EP36478 (3)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 July 2001 (26.07.2001)

PCT

(10) International Publication Number
WO 01/53962 A1

(51) International Patent Classification[7]: G06F 15/16

(21) International Application Number: PCT/US01/02224

(22) International Filing Date: 22 January 2001 (22.01.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/177,394	21 January 2000 (21.01.2000)	US
60/177,395	21 January 2000 (21.01.2000)	US
60/177,396	21 January 2000 (21.01.2000)	US
60/177,397	21 January 2000 (21.01.2000)	US
60/177,398	21 January 2000 (21.01.2000)	US
60/177,399	21 January 2000 (21.01.2000)	US
60/182,434	15 February 2000 (15.02.2000)	US
60/204,386	15 May 2000 (15.05.2000)	US

(71) Applicant (*for all designated States except US*): SORCERON, INC. [US/US]; 75 9th Avenue Floor 6 East, New York, NY 10011 (US).

(72) Inventors; and

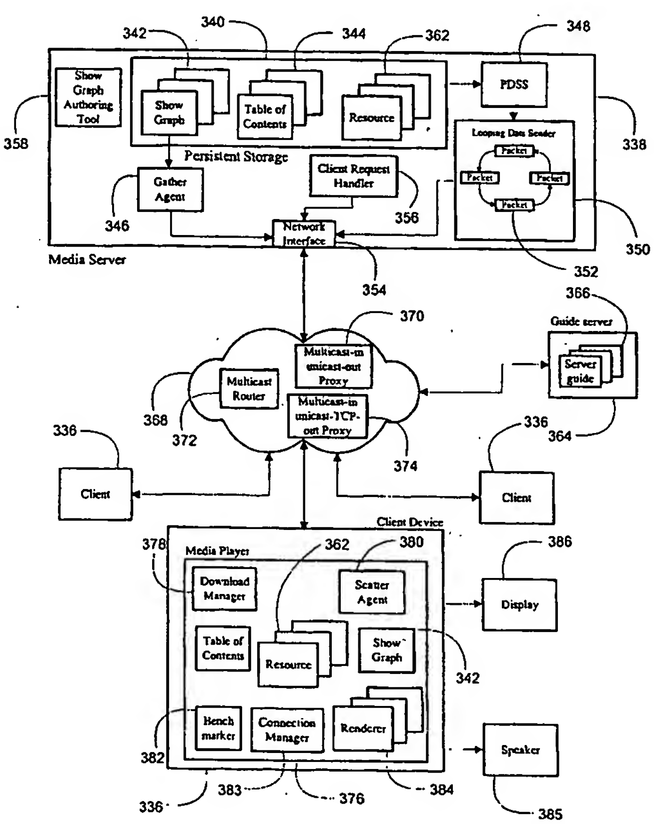
(75) Inventors/Applicants (*for US only*): MCTERNAN, Brennan, J. [US/US]; 135 N. Martine Avenue, Fanwood, NJ 07023 (US). NEMITOFF, Adam [US/US]; 240 Ackerman Avenue, Ridgewood, NJ 07450 (US). MURAT, Altay [US/US]; 104-13 89th Avenue, Richmond Hill, NY 11418 (US). BANGIA, Vishal [IN/US]; 45 River Drive South, Jersey City, NJ 07310 (US). GIANGRASSO, Steven [US/US]; 15 Regent Street, Valley Stream, NY 11580 (US).

(74) Agent: OSTROW, Seth, H.; Brown Raysman Millstein Felder & Steiner LLP, 120 W. 45th Street, New York, NY 10036 (US).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CU, CZ, DE,

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR DELIVERING RICH MEDIA CONTENT OVER A NETWORK



(57) Abstract: Systems and methods are presented that allow the efficient distribution of rich media to clients (336) by maximizing the use of available bandwidth and client processing capabilities. A rich media presentation is divided into discrete components, and a producer of the presentation specifies how a presentation is to be assembled and where resources needed for the presentation are to be found. This information is packaged into a data structure and sent to clients (336). Clients (336) use this data structure to retrieve the necessary resources for the presentation. Producers are able to prioritize the particular resources that form part of the ultimate presentation according to their importance in the presentation, and clients (336) can retrieve the resources most suitable for their capabilities, including processing power, graphics production speed, and bandwidth. A benchmarker routine running on the client (336) helps identify these capabilities just before retrieval of the presentation components, to more closely assess the conditions under which the client (336) will retrieve, assemble and present the desired show.

WO 01/53962 A1

DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEM AND METHOD FOR DELIVERING RICH MEDIA CONTENT OVER A NETWORK

BACKGROUND OF THE INVENTION

The invention disclosed herein relates generally to techniques for distributing multimedia content across computer networks. More particularly, the present invention relates to improved systems and methods for efficiently dividing processing between servers distributing content and clients playing back the received content, thereby allowing a richer experience and maximizing processing power of both clients and servers.

Over the past decade, processing power available to both producers and consumers of multimedia content has increased exponentially. Approximately a decade ago, the transient and persistent memory available to personal computers was measured in kilobytes (8 bits = 1 byte, 1024 bytes = 1 kilobyte) and processing speed was typically in the range of 2 to 16 megahertz. Due to the high cost of personal computers, many institutions opted to utilize "dumb" terminals, which lack all but the most rudimentary processing power, connected to large and prohibitively expensive mainframe computers that "simultaneously" distributed the use of their processing cycles with multiple clients.

Today, transient and persistent memory is typically measured in megabytes and gigabytes, respectively (1,048,576 bytes = 1 megabyte, 1,073,741,824 bytes = 1 gigabyte). Processor speeds have similarly increased, modern processors based on the x86 instruction set are available at speeds up to 1.5 gigahertz (approximately 1000 megahertz = 1 gigahertz). Indeed, processing and storage capacity have increased to the point where personal computers, configured with minimal hardware and software modifications, fulfill roles such as data warehousing, serving, and transformation, tasks that in the past were typically reserved for mainframe computers. Perhaps most importantly, as the power of personal computers has increased, the average cost of ownership has fallen dramatically, providing significant computing power to average consumers.

The past decade has also seen the widespread proliferation of computer networks. With the development of the Internet in the late 1960's followed by a series of inventions in the fields of networking hardware and software, the foundation was set for the rise of networked and distributed computing. Once personal computing power advanced to the point where relatively high speed data communication became available from the desktop, a domino effect was set in motion whereby consumers demanded increased network services,

which in turn spurred the need for more powerful personal computing devices. This also stimulated the industry for Internet Service providers or ISPs, which provide network services to consumers.

Computer networks transfer data according to a variety of protocols, such as UDP (User Datagram Protocol) and TCP (Transport Control Protocol). According to the UDP protocol, the sending computer collects data into an array of memory referred to as a packet. IP address and port information is added to the head of the packet. The address is a numeric identifier that uniquely identifies a computer that is the intended recipient of the packet. A port is a numeric identifier that uniquely identifies a communications connection on the recipient device.

Once the data packet is addressed, it is transmitted from the sending device across a network via a hardware network adapter, where intermediary computers (e.g., routers) relay the packet to the appropriate port on the device with the appropriate unique IP address. When data is transmitted according to the UDP protocol, however, no attempt is made to inform the sender that the data has successfully arrived at the destination device. Moreover, there is neither feedback from the recipient regarding the quality of the transmission nor any guarantee that subsequent data sent out sequentially by the transmitting device will be received in the same sequence by the recipient.

According to the Transmission Control Protocol, or TCP, data is sent using UDP packets, but there is an underlying "handshake" between sender and recipient that ensures a suitable communications connection is available. Furthermore, additional data is added to each packet identifying its order in an overall transmission. After each packet is received, the receiving device transmits acknowledgment of the receipt to the sending device. This allows the sender to verify that each packet of data sent has been received, in the order it was sent, to the receiving device. Both the UDP and TCP protocols have their uses. For most purposes, the use of one protocol over the other is determined by the temporal nature of the data. Data can be viewed as being divided into two types, transient or persistent, based on the amount of time that the data is useful.

Transient data is data that is useful for relatively short periods of time. For example, a television transmits a video signal consisting of 30 frames of imagery each second. Thus, each frame is useful for $1/30^{th}$ of a second. For most applications, the loss of one frame would not diminish the utility of the overall stream of images. Persistent data, by

contrast, is useful for much longer periods of time and must typically be transmitted completely and without errors. For example, a downloaded record of a bank transaction is a permanent change in the status of the account and is necessary to compute the overall account balance. Loosing a bank transaction or receiving a record of a transaction containing errors would have harmful side effects, such as inaccurately calculating the total balance of the account.

UDP is useful for the transmission of transient data, where the sender does not need to be delayed verifying the receipt of each packet of data. In the above example, a television broadcaster would incur an enormous amount of overhead if it were required to verify that each frame of video transmitted has been successfully received by each of the millions of televisions tuned into the signal. Indeed, it is inconsequential to the individual television viewer that one or even a handful of frames have been dropped out of an entire transmission. TCP, conversely, is useful for the transmission of persistent data where the failure to receive every packet transmitted is of great consequence.

One of the reasons that the Internet is a successful medium for transmitting data is because the storage of information regarding identity and location of devices connected to it is decentralized. Knowledge regarding where a device resides on a particular part of the network is distributed over a plurality of sources across the world. A connection between to remotely located devices can traverse a variety of paths such that if one path becomes unavailable, another route is utilized.

The most simplistic path is between two devices located within a common Local Area Network, or LAN. Because these devices are located on a common network, they are said to reside in the same subnet. Each network on the Internet is uniquely identified with a numeric address. Each device within a network, in turn, is identified by an IP address that is comprised of a subnet address coupled with a unique device ID. According to version four of this standard ("IPv4") an IP address is a 32-bit number that is represented by four "dot" separated values in the range from 0 through 255, e.g., 123.32.65.72. Each device is further configured with a subnet mask. The mask determines which bits of a device's IP address represent the subnet and which represent the device's ID. For example, a device with an IP address of 123.32.65.72 and a subnet mask of 255.255.255.0 has a subnet address of 123.32.65 and an ID of 72.

Each packet of data sent by a device, whether it is formatted according to the UDP or TCP protocols, has a header data field. The header is an array of bytes at the beginning of a packet that describe the data's destination, its origin, its size, etc. When a sender and recipient are both located within the same subnet, the recipient device's network hardware examines network traffic for packets tagged with its address. When a packet addressed to the recipient is identified, the network hardware will pass the received data off to the operating system's network services software for processing.

When a sender and recipient are located in different subnets, data is relayed from the originating subnet to the destination subnet primarily through the use of routers. While other physical transport methodologies are available, e.g., circuit switched transmission systems such as ATM (Asynchronous Transfer Mode), the majority of computer networks utilize packet switched hardware such as routers. A router is a device that interconnects two networks and contains multiple network hardware connections. Each network connection is associated with, and provides a connection to, a distinct subnet.

Two tasks are performed when a packet, destined for a subnet that is different from the subnet it is currently in, reaches a router within the current subnet. First, the router will examine the subnets that it is connected to via its network hardware. If the router is connected to the packet's destination subnet, it forwards the packet to the router in the appropriate subnet. If the router is not directly connected to the packet's destination subnet, it will query other routers available on its existing connections to determine if any of them are directly connected to the destination subnet. When a router directly connected to the destination subnet is discovered, the packet will be forwarded to it. Where a router connected to the destination subnet is not found, however, the router will propagate the packet to a top level router that is strategically placed to allow access, either directly or through other top level routers, to the entire Internet. These top level routers are currently maintained by a registration authority under government oversight.

The transmission method described above is referred to as the unicast method of transmission, whereby a sender establishes a unique connection with each recipient. By utilizing a unicast model, the specific address of the receiving machine is placed in the packet header. Routers detect this address and forward the packet so that it ultimately reaches its intended recipient. This method, however, is not the most efficient means for distributing

information simultaneously to multiple recipients. The transmission method that best facilitates broadcasting to many recipients simultaneously is multicasting.

Multicasting relies on the use of specialized routers referred to as multicast routers. These routers look only for data packets addressed to devices in the range of 224.0.0.0 through 239.255.255.255. This address range has been specifically set aside for the purpose of facilitating multicast transmissions. Multicast routers retain an index of devices that wish to receive packets addressed to ports in this address range. Recipients wishing to receive multicast packets "subscribe" to a specific IP address and port within the multicast address space. The multicast routers respond to the subscription request and proceed to forward packets destined to the particular multicast address to clients who have subscribed to receive them.

Under the multicast model, the sender transmits packets to a single address, as opposed to the unicast model where the data is transmitted individually to each subscribing recipient. The multicast routers handle replication and distribution of packets to each subscribing client. The multicast model, like the broadcast model, can be conceptually viewed as a "one-to-many" connection and, therefore, must use the UDP protocol. UDP must be utilized because the TCP protocol requires a dialog between the sender and receiver that is not present in a multicast environment.

Clearly, there have been drastic improvements in the computer technology available to consumers of content and in the delivery systems for distributing such content. There have also been drastic improvements in bandwidth available for transmission of video and other multimedia, such as through the use of broadband systems such as digital subscriber lines and cable systems employing cable modems.

However, such improvements have not been properly leveraged to improve the quality and speed of video distribution. For example, existing multimedia distribution systems over the Internet offer the content to all or to classes of clients on the same basis, much as is done in traditional analog systems such as television broadcasting. As a result, these systems fail to take account of the wide variability in processing, storage and other technical capabilities among personal computers.

There is thus a need for a system and method that takes account of such client variability, and that furthermore distributes responsibilities for video distribution and presentation among various components in a computer network to more effectively and

efficiently leverage the capabilities of each part of the network and improve overall performance.

Current methods of video compression use much bandwidth yet provide small, low resolution images and low frame rates per second. Indeed, current video transmission technologies for distribution of video over computer networks such as the Internet attempt to treat the network as an electromagnetic medium, the medium used for broadcasting of television signals. For example, as shown in Fig. 20, a video produced for distribution over the Internet consists of a scene 10, which may have a set 12 and one or more live actors 14, recorded by a camera 16. The scene is recorded as a series of two-dimensional images 18 which are compressed and transmitted such as by streaming or multicasting to a client device 20. The resulting video is presented on the client device 20 as a small image having low resolution and fewer frames per second than a standard broadcast television video signal. The resulting video is thus lacking substantially in quality as compared to typical television signals to which consumers are accustomed.

Broadband technologies such as fiber optic lines, cable systems and cable modems, satellite transmission systems, and digital subscriber lines promise to improve the situation by increasing bandwidth substantially. However, even the increased level of bandwidth provided in broadband systems may not be sufficient for many applications, such as the distribution and display of multiple simultaneous video signals used, for example, in teleconferencing applications. Furthermore, broadband technologies will not be in widespread usage for quite some time. It is also likely that video distribution technology will continue to push and exceed the limits of the transmission system capable of carrying the signals, including broadband systems.

There is thus a need for improved systems and methods for distributing video signals which require lower bandwidth but provide improved display size and resolution.

BRIEF SUMMARY OF THE INVENTION

It is an object of the present invention to solve the problems described above relating to existing content delivery systems.

It is another object of the present invention to alter the traditional model of distribution of video and other multimedia content to take advantage of each particular client's capabilities.

It is another object of the present invention to make the distribution of video and other multimedia content more efficient and effective through a better distribution of tasks.

It is another object of the present invention to provide more efficient retrieval of content from servers.

It is another object of the present invention to allow multiple clients to retrieve the same content from the same server while minimizing interruptions of the server delivering the content.

It is another object of the present invention to provide clients the ability to select the best connection available to them for the delivery of content.

It is another object of the present invention to more effectively manage the process of connecting to servers over the Internet.

It is another object of the present invention to reduce the amount of bandwidth required to deliver a video signal across a computer network.

It is another object of the present invention to so reduce the bandwidth while still improving the quality of the video transmission.

It is another object of the present invention to increase resolution of video images distributed over a computer network.

It is another object of the present invention to increase the size of a video display distributed over a computer network.

Some of the above and other objects are achieved by a system and method that allows the efficient distribution of rich media to clients by maximizing the use of available bandwidth and client processing capabilities.. Rich media refers generally to multiple types of digital media that are directly sensed by a viewer, including video, audio, text, graphics, and the combination of these and other media. In accordance with the invention, a rich media presentation is divided into discrete components, and a producer specifies how a presentation is to be assembled and where resources needed for the presentation are to be found. This information is packaged into a data structure and sent to clients. Clients use this data structure to retrieve the necessary resources for the presentation.

This modularization of the presentation provides numerous advantages. For example, producers are able to prioritize the particular resources that form part of the ultimate presentation according to their importance in the presentation. It also allows clients to

retrieve the resources most suitable for their capabilities, including processing power, graphics production speed, and bandwidth. A benchmarker routine running on the client helps identify these capabilities just before retrieval of the presentation components, to more closely assess the conditions under which the client will retrieve, assemble and present the desired show.

In preferred embodiments, the client device works in a highly autonomous manner, thereby allowing the server to use multicast techniques to distribute data to many clients simultaneously. This autonomy allows each client the ability to display received rich media in the most effective way allowed by each individual client's hardware profile.

Some of the objects of the present invention are achieved through a method for preparing a multimedia presentation for transmission to a client over a network. The method involves allowing a producer of the presentation to identify elements of software which process data representing resources used in the presentation, specify connections between two or more elements, wherein a connection represents a flow of data from one element to another element, and specify a plurality of resources to be processed by the identified elements and location data indicating the locations of the resources on one or more servers connectable to the network. The method further involves generating a set of presentation data structures, including the identified elements, connections, and resources, for transmission to a client. This enables the client to reproduce the multimedia presentation from the presentation data structures by retrieving at least some of the resources and processing the retrieved resources with the identified elements in accordance with the specified connections.

The set of presentation data structures may include a show graph representing a plurality of identified elements and the connections extending therebetween. The presentation data structures may also include a table of contents listing all or some of the specified resources and corresponding locations. The producer may also be able to replace or swap a first identified element with a second element while retaining for the second element any connection and resources associated with the first element.

As another advantage of the present invention, producers may further specify an encoder to tap a signal on a selected connection and encode the data flowing at the selected connection. The producer may associate the specified encoder with a given set of client processing capabilities or available bandwidth for transmission to a client. Thus, if the server receives data indicating the processing capabilities of a client requesting the presentation or

available bandwidth for transmission of the presentation to the client, such as from a benchmarking program running on the client, an agent on the server can traverse a series of identified elements via their connections until a tapped encoder is located being associated with the client's processing capabilities or available bandwidth.

The data flowing at the tapped connection, e.g., the data output from the element at the start of the connection, is then encoded using the specified encoder, and made available for transmission to the client. When the client receives the presentation data structures, it executes an agent which traverses the series of identified elements via their connections until a tapped decoder is located being associated with the client's processing capabilities or available bandwidth. The agent initiates this decoder to decode the data at the tapped connection.

Objects of the present invention are also achieved by a system for delivering a multimedia presentation from a server to a client. The system contains a presentation authoring tool for use by a producer of the presentation to identify software elements for processing data representing resources used in the presentation, specify connections between two or more elements representing a flow of data from one element to another element, specify a plurality of resources to be processed by the identified elements and location data indicating the locations of the resources on one or more servers connectable to the network, and specify an encoder to tap a signal on a selected connection and encode the data flowing at the selected connection. The system also contains an agent for traversing a series of identified elements via their connections until a tapped encoder is located and encoding the data at the tapped connection using the specified encoder.

Some of the above and other objects are further achieved by a computer implemented method for receiving content data transmitted from a server in a sequence of packets, where the server repeatedly transmits the packets in sequence or loops through the sequence. The method involves, upon a client's receipt of a first packet from the server, deriving from the packet a number of the packet in the sequence and a total number of packets in the sequence. The server inserts this data into the header of the packet before transmission. The client then generates and stores an index having an entry for each of the packets in the sequence based upon the total number of packets in the sequence.

The client may further allocate a buffer in memory for storing the expected packets, with the size of the buffer being determined by multiplying the size of the first

packet by the total number of packets. The client may determine the size of the first packet by reading such data from the header, if inserted there by the server, or measuring the packet. The allocated buffer space is exactly the required amount if the server broke the content into equal sized packets. Otherwise, the buffer represents approximately the amount of memory needed. Alternatively, if non-equal sized packets are used, the total data size of all packets may be stored in the header of each packet as well, so that a buffer with an appropriately allocated amount of space may be provided by the client.

The method further involves the client updating the index for each packet received subsequent to the first packet, by registering the received packet's number in the index. The client may further store the packet data in the allocated buffer space at the proper location in the sequence. The client uses the index to detect whether any subsequent packet is missing from the sequence of packets. This may be done on the fly, by detecting whenever a subsequent packet is received whether the prior packet just received precedes the current packet consecutively in the sequence, or may be done after the sequence has begun to repeat packets.

If a missing packet is detected, the client determines whether the first time required to retrieve the missing packet by waiting for the packet to be received in the repeating sequence is greater than a threshold time. If the first time is greater than the threshold time, the client requests the missing packet to be delivered from a server. If the first time does not exceed the threshold, the client waits for the sequence to loop around again to the missing packet, and then receives and stores the missing packet. The threshold time may be a predefined time set by a producer of the content or a server administrator and included in a software application executing on the client and performing this process, or may be computed as the time required to request and receive the missing packet from the server based, for example, on available bandwidth.

Objects of the invention are also achieved by a system for delivering content from a server to one or more clients such as over the Internet. The system includes a server, such as a multicast server, for transmitting an item of content in a sequence of packets, each packet containing a header storing a number of the packet in the sequence, and the packets being transmitted as repeating loops of the packets in sequence. The system also includes a client system for subscribing to the multicast server, receiving the transmitted packets, tracking the receipt of packets using the packet numbers, identifying any packets in the

sequence which are missing based on the tracked packet numbers, and deciding whether to wait for any given missing packet to be received in the subsequent loop or request the missing packet from the server. The server transmits the missing packet in response to a request received from the client system. The server may include a packetized data source structure for decomposing the content into the sequence of packets.

The above and other objects are achieved by a software component running on a client computer connected to a network such as the Internet which manages the connection of the client to a server to receive the delivery of content. When a client requests an item of multimedia content, such as a program, movie, or other video or audio file, the connection manager software retrieves a server guide over the Internet from a guide server. The server guide lists the servers, e.g., by server address and server type, from which the content may be retrieved. In some embodiments, the guide server stores a number of different server guides representing different locations at which the content may be accessed, and selects one of the server guides based on load balancing, resource allocation, or other concerns.

The server guide lists servers using different mechanisms or types of transmissions. Such types include servers configured to multicast content, servers configured to receive a multicast transmission and package the data for unicast transmission using, e.g., UDP, and servers configured to receive a multicast transmission and package data for TCP transmission. Preferably, the server guide lists the servers in a desired connection sequence, such as multicast router, multicast-in unicast-out proxy, and then multicast-in unicast-TCP-out proxy.

Some of the above and other objects are also achieved by a method for managing a retrieval of multimedia content over a computerized network, the network having a plurality of servers connectable to one or more clients. The method involves retrieving at a first client a server guide identifying a list of servers capable of delivering a selected item of multimedia content and the first client automatically determining whether a connection may be made to a first server identified in the server guide to achieve delivery of the selected content item. If the connection may be made, the first client establishes a connection with the first server to retrieve the selected content item therefrom. If the connection is unable to be made, the first client automatically determines whether a connection may be made to a second server identified in the server guide to achieve delivery of the selected content item. The first client repeats these steps for the second server and any additional server(s) identified in the

server guide until a connection may be made to a server by which the selected content item may be delivered.

The servers identified in the server guide may include one or more routers connectable to a content server, the content server storing the selected content item. In some embodiments, the first server is a multicast router and the second server is a multicast-in unicast-out proxy configured to receive data from the multicast router and provide a unicast connection to the first client such as via UDP. In addition, the server guide may identify as a third server a multicast-in unicast-TCP-out proxy configured to receive requests for parts of the content item from clients, subscribe to the multicast router or multicast-in unicast-out proxy router, and deliver to clients data packets via, e.g., TCP, representing requested parts of the content item. The steps of automatically determining whether a connection may be made are preferably performed first for the multicast router, then for the multicast-in unicast-out proxy router, and then for the multicast UDP router, but may be performed in any given sequence provided in the server guide.

Some of the above and other objects of the present invention are also achieved by a system for establishing a connection over a network to retrieve multimedia content. The system contains a memory device storing a server guide identifying a list of servers capable of delivering a selected item of multimedia content, the list including servers differing in transmission techniques. The server guide may have been downloaded from a server which provides such guides. The system further contains a connection manager for automatically attempting to establish a connection to the servers contained in the list one at a time and, upon determining that a connection can not be established for a given server, attempting to establish a connection to another server in the list until a connection is established or connections can not be established to all servers.

Objects of the invention are also achieved by distributing between a server and client the effort required to create imagery on a client device. The server sends the client three general types of data - a three-dimensional model of a virtual set, compressed video of action occurring, and positional data representing the position and orientation of the camera. The virtual set represents a relatively static environment in which different actions may occur, while the video represents a series of images changing over time, such as person talking, running, or dancing, or any other item or actor undergoing movement. The positional data

allows for the proper orientation of the 3D set consistent with a given view of the action in the video.

Advantageously, the server may send one or more 3D virtual sets well in advance of any given video, and the client can store the model of the virtual set in persistent memory and can use the model with an ongoing video stream and reuse it with later video signals. This reduces the bandwidth required during transmission of the video. Additional identification data may be transmitted with a given video to associate it with a previously transmitted virtual set.

The client receiving these data items compiles them to produce a presentation. The video of the action is rendered onto two-dimensional images of the stored virtual set, such as by texture mapping, at a predefined location within the set at which the action would have occurred if done on a corresponding real set. For example, if the set is a backdrop for a news broadcast, and the video is of a person reporting the news, the video is placed at a location within the set in which the person would have sat while reporting the news. Additional video or other multimedia content may be transmitted, received and positioned at other locations within the virtual set, such as on boards behind the news reporter, using the same or similar techniques.

The video may be live action recorded by cameras or virtual action produced through the use of computer graphics. To improve performance, the video of the action is processed and compressed prior to transmission. In one embodiment, the video is matted to produce a high contrast image such as in black and white, with the white region identifying the portion of the video representing the action and the black region representing inactive portion of the video such as the background. When the video is recorded with cameras, the actor is placed before a blue screen for the filming. The video of the actor is processed with systems well known in the art that can generate a high contrast image where the white part of the image represents the area occupied by the actor and the black part of the image represents the area occupied by the blue screen. The high contrast image is then overlaid on the video to identify the active areas of the video. The video is cropped to eliminate as much of the inactive regions as practical or possible, with the remaining black, inactive portions being made transparent for overlaying on the rendered image of the virtual set.

The positional data indicates where the real camera is in relation to actor on the real set. This data is used to position the 3D Camera in the 3D set. Because the 3D

camera's position and orientation match that of the camera that captured the video, the video retains its dimensionality. Some of the above and other objects of the present invention are achieved by a method for distributing video over a network for display on a client device. The method includes storing model data representing a set in which action occurs, generating video data representing action occurring, capturing positional data representing a position of one or more actors during the action in the generated video, and transmitting from a server to the client device as separate data items the model data, generated video, and positional data, to thereby enable the client to reproduce and display a video comprising the action occurring at certain positions within the set.

Some of the above and other objects of the present invention are achieved by method for receiving video over a network and presenting it on a client device. The method includes receiving from a server as separate data items model data representing a set in which action occurs, video data representing action occurring, and positional data representing a position of one or more actors during the action in the generated video. The method further involves rendering the video data within the set at a predefined position within the set determined at the time the virtual set was constructed, and presenting the video on a client device.

Objects of the invention are also achieved through a system for preparing a video for distribution over a network to one or more clients, the video containing one or more actors. The system contains a positional data capturing system for capturing position data representing a position of the camera relative to the actors in the video, a video compression system for reducing the video by eliminating all or a portion of the video not containing the actor, the video compression system including a matting system for matting the video to separate the actor from other parts of the video, and a transmission system for transmitting compressed video in association with corresponding positional data and in association with model data representing a set within which the video is rendered for presentation by one or more clients.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is illustrated in the figures of the accompanying drawings which are meant to be exemplary and not limiting, in which like references are intended to refer to like or corresponding parts, and in which:

Fig. 1 is a block diagram presenting data flow and distribution according to one embodiment of the present invention;

Fig. 2 is a block diagram presenting the distribution of data packets upon receipt by a client device, according to one embodiment of the present invention;

Fig. 3 is a block diagram presenting the configuration of various hardware and software components according to one embodiment of the present invention;

Fig.4 is a flow diagram presenting the process of looping distribution of data, according to one embodiment of the present invention;

Fig. 5 is a flow diagram continuing the process of looping distribution of data, according to one embodiment of the present invention;

Fig. 6 is a block diagram presenting the decomposition of a segment of data into discrete packets, distribution of the packets, and concatenation of the packets into the original data segment by the client, according to one embodiment of the present invention;

Fig. 7 is a flow diagram presenting an overview of the connection management process according to one embodiment of the present invention;

Fig. 8 is a flow diagram presenting the process of connection management using various proxy servers, according to one embodiment of the present invention;

Fig. 9 is a block diagram presenting a multicast client connecting to a server via a network, according to one embodiment of the present invention;

Fig. 10 is a block diagram presenting a non-multicast enabled client connecting to a server via a network, according to one embodiment of the present invention;

Fig. 11 is a block diagram presenting a client capable of initiating only TCP connections connecting to a server via a network, according to one embodiment of the present invention;

Fig. 12 is a block diagram presenting the layout of software elements within a Show Graph Authoring Tool according to one embodiment of the present invention;

Fig. 13 is a block diagram presenting the interconnection of software elements within the Show Graph Authoring Tool according to one embodiment of the present invention;

Fig. 14 is a flow diagram presenting the distribution and utilization of the Show Graph according to one embodiment of the present invention;

Fig. 15 is a block diagram presenting the inclusion of a low bandwidth tap within a Show Graph, displayed by the Show Graph Authoring Tool, according to one embodiment of the present invention;

Fig. 16 is a block diagram presenting the inclusion of a high bandwidth tap within a Show Graph, displayed by the Show Graph Authoring Tool, according to one embodiment of the present invention;

Fig. 17 is a block diagram presenting the inclusion of both low and high bandwidth taps within a Show Graph, displayed by the Show Graph Authoring Tool, according to one embodiment of the present invention;

Fig. 18 is a flow diagram presenting the distribution and utilization of a Show Graph that includes taps, according to one embodiment of the present invention;

Fig. 19 is a flow diagram presenting the process of client benchmarking according to one embodiment of the present invention;

Fig. 20 is a flow diagram showing the prior art method for recording and distributing video over a network;

Fig. 21 is a block diagram of a system implementing one embodiment of the present invention;

Fig. 22 is a flow chart showing a process of generating and distributing video in the system of Fig. 21 in accordance with one embodiment of the present invention;

Fig. 23 is a flow diagram showing components and processes involved in the process shown in Fig. 22; and

Fig. 24 is a diagram illustrating triangulation of marker positions in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Embodiments of the present invention are now described with reference to the drawings in Figs. 1-24. Fig. 1 presents a conceptual overview of embodiments of the invention involving creation, synchronization and distribution of a data stream to clients across a computer network. The data for a rich media presentation or show is acquired from various sources 302. The presentation data is comprised of a number of media types presented in an integrated fashion. Exemplary media types include, but are not limited to, motion data, camera position data, audio data, texture map data, polygon data, etc. Components of the system described more fully below generate data packets 304 for each

media type and load or transferred them to a server computer 306. Each packet of data is synchronized with one or more other packets to display a coherent presentation. For example, a plurality of video frames comprises motion data, audio data and text data. Each packet of data contains information regarding a particular type of media for a specific frame of the video. If the packets are not synchronized, the audio and visual portions of the video will not be presented in their intended order, rendering the receiving device unable to present a coherent scene to the viewer. The server 306 receives the data packets and assembles them into a data stream 308.

The data stream 308 is distributed to clients 314 across a computer network 310. According to one embodiment, the data is transmitted to a specific port on a multicast router 312 residing on the Internet, an intranet or other closed or organizational network 310. Multicast routers execute multicast daemon software (not shown) that accepts subscriptions from clients that wish to receive multicast data sent to the port subscribed to. When the multicast router receives data packets 304, such as the data packets contained in a data stream 308, each packet is duplicated and transmitted simultaneously to all subscribing clients 314. This methodology allows multiple clients 314 to receive a desired data stream 308 without requiring the server 306 to maintain and manage a connection with each individual client.

As described more fully below, the breakdown of the show into discrete components allows for a number of enhancements to reduce bandwidth, increase efficiency, and improve the ultimate quality of the show on the client computers. Different renderers process the different media types. Producers adjust settings within the show. Clients use renderers differently in accordance with client processing capabilities or available bandwidth.

Fig. 2 illustrates how, in presenting a show, the client breaks up packets from the server. The client 336 receives a data stream 328 comprised of a number of coordinated media packets 319 and 321. A distributor 330 parses the data stream and decomposes it back into its constituent packets 322. The distributor 330 may be embodied in hardware and software, or may be implemented as a software program executing on the client. The client stores or has access to a number of renderers 334. Each renderer 334 is configured to accept packets 322 of a particular media type and convert it into information that may be experienced by the viewer. Each packet 322 is rendered by its associated renderer 334 and presented to the viewer. For example, an audio renderer takes packets of audio data and

generates sound information heard by the listener while a video renderer accepts packets of video data and presents it to the viewer on a display device.

Referring to Fig. 3, a system of one preferred embodiment of the invention is presented. A number of clients 336 and servers 338 and 364 are connectable to a network 368 by various means. For example, if the network 368 is the Internet, the servers 338 and 364 may be web servers that receive requests for data from clients 336 via HTTP, retrieve the requested data, and deliver them to the client 336 over the network 368. The transfer may be through TCP or UDP, and data transmitted from the server may be unicast to requesting clients or multicast to multiple clients at once through a multicast router.

In accordance with the invention, the Media Server 338 contains several components or systems including a Show Graph Authoring Tool 358, a Gather Agent 346, a Packetized Data Source Structure 348, a Looping Data Sender 350, and a Client Request Handler 356. The Media Server 338 further contains persistent storage 340, such as a fixed hard disk drive, to store data including show graphs 342, tables of contents 344, and show resources 362. These components and data may be comprised of hardware and software elements, or may be implemented as software programs residing and executing on a general purpose computer and which cause the computer to perform the functions described in greater detail below.

The Packetized Data Source Structure 348 is provided to retrieve data from any number of data sources, such as a persistent storage device 340. Data producers use the data source, such as a conventional database, to manage media content such as video, audio, graphics, or text content. The database may be a relational database, an object-oriented database, a hybrid relational object oriented database, or a flat-file database. In other embodiments, the data store is simply a persistent storage device, such as a fixed hard disk, with a file system managed by the server's OS. Data selected from the data store for transmission is placed in a data buffer, which provides transient storage for data that is to be manipulated prior to transmission.

The Packetized Data Source Structure 208 retrieves data temporarily held within the buffer. The Packetized Data Source Structure 348 takes a contiguous segment of data as input and produces a plurality of discrete data packets 352. Each data packet 352 is tagged with identifying information including the packet's position or number in the overall sequence of packets, the total number of packets that comprise the contiguous portion of data

that is to be sent to the client, and the number of bytes contained within the packet. The packets 352 may be tagged with additional information required by a given communications protocol, hardware device, or software application requesting the data on the client device.

One or more Looping Data Senders 350 receive packets 352 generated by the Packetized Data Source Structure 348. The Looping Data Sender 350 takes each packet 352 and transmits it by way of an integrated or external network interface 354 to clients 336 via a network 368. After the final packet 352 in the sequence is received and transmitted, the Looping Data Sender 350 begins re-transmitting the packets starting with the first packet in the sequence. In this manner, the Looping Data Sender continually "loops" through the transmission of the packets, allowing clients 336 to receive all packets regardless of the point in the transmission sequence at which they began receiving packets. This allows clients 336 to receive any dropped or otherwise missing packets without having to interrupt the server 338 and 364 or waste bandwidth transmitting requests for retransmission. Among other things, the use of this delivery mechanism in the show distribution system described herein enlarges the number of connections that can be made to receive a show at any one time.

The server 338 or 364 transmits packetized data via the network 368 to any client 336 requesting the data. The client is equipped with an integrated or external network adapter used to receive data packets from the network 368. The client has persistent and transient memory for storing received data packets, storing and executing application programs, and storing other resources. One application stored in persistent memory and executed in transient memory by the client is a Media Player 376, which is used for the playback of multiple types of media including, but not limited to, audio, video, and interactive content.

The Media Player 376 contains several components or systems including a Download Manager 378, a Scatter Agent 380, a Benchmarker 382, a Connection Manager 383, and one or more Renderers 384. In alternative embodiments, these components are standalone software or hardware components accessed by executing applications, such as the Media Player 376.

The Media Player 376 issues requests for media packets 352 to a server 338 or 364. If the server 338 or 364 is multicasting the content, the client request takes the form of a subscription to the router 372. Packets are received across the network 368 via the client's network interface adapter. The Media Player 376 or other application requesting data from

the server accepts and records receipt of packets in memory. Upon receipt of a duplicate packet, the client will stop receiving further packets, as the receipt of a duplicate packet is an indication that the packet sequence has looped around to the point at which the client first starting receiving packets and therefore the client should have received al the packets in the sequence. The client checks whether any packets in the sequence are missing and, if so, determines if the time to wait for the Looping Data Sender 350 to retransmit the packet is greater than a time threshold, such as the time needed to directly request and receive the missing packet or packets from the server, or a predefined threshold set by the content producer. If the time to wait for the packet to be received is greater than the threshold, the Download Manager 378 issues a request to the Client Request Handler 356. Upon receiving the request, the Client Request Handler 356 accesses the Looping Data Sender 350, duplicates the requested packet and transmits it to the client 336. The result is that clients are continually fed a stream of requested data and can recover missing packets by either simply awaiting retransmission of the packet or requesting it directly, whichever the client deems is most efficient given the bandwidth constraints of the client.

The operations of the Packetized Data Source Structure and Looping Data Sender are described in greater detail in Figs. 4-6. Referring to Fig. 4, data to be transmitted to clients is extracted from a data source and placed within a data buffer for temporary storage prior to distribution, step 226. The Packetized Data Source Structure periodically fetches data from the buffer as a single contiguous segment of data, step 228. The retrieved data is decomposed into a plurality of discrete data packets optimized for transmission across the type of network the computer executing the software is connected to, step 230. The packets may be of equal size to one another, or may vary slightly in size as desired to optimize them for transmission. Furthermore, the data may be of any type or format, due to the fact that the Packetized Data Source Structure breaks a large portion of data into a series of smaller pieces and makes no substantive analysis of the data it is decomposing.

Each packet consists of data and a header structure. The Packetized Data Source Structure tags each packet header with a unique packet identifier identifying the packet or piece of data being referenced, the number of bytes comprising the packet, followed by the actual bytes of data themselves, step 232. The Packetized Data Source Structure also labels each packet with the total number of packets in the sequence, allowing the client to determine how many packets are expected and which packets are missing from a

transmission. The Packetized Data Source Structure holds the sequence of packets until transmission.

The packets are sent through one or more Looping Data Senders. The Looping Data Sender retrieves the next packet in the sequence held by the Packetized Data Source Structure and transmits it to the client, step 234. In embodiments where the client is receiving data via a multicast router, the Looping Data Sender transmits the packets to the multicast address, which handle duplication and transmission of the data to all subscribing clients. In Unicast embodiments, the Looping Data sender transmits data directly to the requesting client. The Looping data sender continues to fetch each data packet in the sequence and loops around to start transmission at the first packet in the sequence after all packets have been transmitted, step 234.

Clients receive data transmitted across a network from the Looping Data Sender. When the first packet is received, the client examines its header data to determine the packet size, the total packet count for the transmission, and the packet number of the first packet, step 236. The total transmission length can be determined by this data, e.g., by multiplying the size of the received packet by the total number of packets in the transmission. A buffer capable of storing at least the total transmission length is opened in memory on the client to temporarily store the packets, step 238, before being acted upon by a playback engine or other software by which the data is processed.

The client creates a table or index to record the reception status for each expected packet received, step 240. The index is assigned a number of entries equal to the total number of packets in the sequence. The number of the first packet is recorded as received in the index, and a pointer is moved in the index to the next expected packet in the sequence.

As the client receives data packets from the server, the reception status of each expected packet is recorded in the index created on the client device, step 242. The packet data is stored at the appropriate place in the buffer. The client continues to receive data packets and to record which packets have been received. The packet number extracted from the packet's header determines the storage location within the index where it will be placed. Packets continue to be received until a packet that is already recorded in the client index is received, step 244. When a duplicate packet has been received, a check is performed to determine if all expected packets have been received, e.g., the client examines its index to

determine if it is complete or if index entries are missing, step 246. If all expected packets have been received, the transmission is complete and processing ends, step 248. The client now has the complete set of data and is free to manipulate it with the software application the data was intended for.

When transmitting data across a network, it is possible that one or more packets will be lost or "dropped" during transmission. Turning to Fig. 5, processing continues where an expected packet or packets forming the total transmission is not received. The client will determine the packet number of the last packet received and set it as the current packet, step 250. The client will also determine the packet identifier for the missing packet that is furthest from the current packet, step 252. Although a number of other packets may be missing, the client should only have to wait until the furthest such missing packet is retransmitted. Using this information as inputs, the client calculates the time it will take for the Looping Data Sender to retransmit the missing packet or packets based upon the bandwidth available for the transmission, step 256.

The Download Manager uses the calculated time that it will take to receive any missing packet if broadcast in its regular sequence from the Looping Data Sender and compares it with the time threshold, step 258. According to one embodiment, the threshold is a pre-determined value set in the download manager or the software application that is expecting and will act upon the received data. Alternatively, in other embodiments, the threshold is dynamically set to the time the Download Manager calculates it will take to directly download the packet from the server, bypassing the normal sequence in which the Looping Data Sender transmits the packets. This calculation can be a function of the existing bandwidth currently available based, for example, on currently experienced transmission times.

The Download Manager takes one of two different actions based on whether the time to await transmission of the missing packet from the Looping Data Sender is greater than the threshold, step 260. If the time to await transmission of the missing packets by the Looping Data Sender is less than the threshold, the client simply waits until the Looping Data Sender re-transmits the missing packet or packets and the routine ends, steps 262 and 268. Where the time to await retransmission is greater than the threshold, step 264, the client instructs the Download Manager to initiate a direct connection with the server via the Client Request Handler. The Download Manager transmits the index number of the missing packet

or packets to the Client Request Handler, which, in turn, duplicates the desired packets from the Looping Data Sender and transmit them directly to the client. Upon receipt by the client, the packet's status is recorded in the index and processing is complete, step 268.

As an alternative, a missing packet may be detected by noting a skipped entry in the index following receipt of any given packet in the sequence. The download manager can then determine whether it should wait for the sequence to loop around again or specifically request the missing packet.

Because interaction required between the server and client to download data is eliminated or greatly reduced, download speeds are improved. Clients can "listen" to a server looping data and receive the required information without interrupting the server to get it. In this manner, bottlenecks such as transmitting a request for data and waiting for the server to respond are eliminated. Thus, each bandwidth purchaser takes full advantage of the specific bandwidth available.

Fig. 6 illustrates the process of sending and receiving looping data as described herein. A server 270 retrieves a contiguous segment of data 272 from a data source and places it in a buffer. Data contained in the buffer, in this instance text data, is passed through a Packetized Data Source Structure 274 where the data is spilt into a plurality of packets and modified to include the packet's unique numeric id and data indicating the total number of packets in the sequence. A Looping Data Sender 278 retrieves each packet 276 and transmits it to a multicast address located on a network 280 for distribution to subscribing clients 282. When the Looping Data Sender transmits the final packet in the sequence, it begins retransmission with the first packet.

The client 282 subscribes to a multicast address to receive the data packets. The first packet is received and placed in an index 284 created on and stored by the client 282. According to this illustration, the client first receives packet number six in the sequence. Because each packet contains data indicating the total number of packets in the sequence, the index is adjusted accordingly. Since each packet is transmitted in sequence, the client receives packet seven, followed by packets one through five. After receiving packet five, the next packet is recognized as a duplicate and reception of additional packets is halted. The client checks to determine if all packets in the sequence have been received. When the entire sequence is received, the client will concatenate the separate packets into the same contiguous segment of data that is stored on the server 286.

Returning to Fig. 3, client device 336 contains and executes a Connection Manager 383 in order to negotiate and maintain a connection with Media Servers 338. The Connection Manager 383 executes routines on the client 336 when an attempt is made to establish the connection. As explained more fully below, the routines include directing the client 336 to establish a multicast, multicast-to-unicast, or unicast TCP connection based upon the requirements of the network provider that the client device 336 is using to connect to the data network 368. The connection manager software 383 further determines appropriate bandwidth and ensures that resources are being received appropriately.

When a client 336 requests the transmission of content, a connection is first established with a Guide Server 364. The Guide Server 364 parses the client request and returns an appropriate Server Guide 366 based on the request. The Server Guide 366 comprises a listing of all Media Servers 338 connected to the network 368 that are capable of transmitting the content requested by the client 336 via its Connection Manager 383. The client 336 receives the Server Guide 366 and attempts to initiate a connection with the first Media Server entry in the guide 366. The Connection Manager 383 opens a connection between a Media Server 338 and the client 336 based on the server address listed in the Server Guide 366. The client 336 attempts to make a connection with the Media Server 338 using the servers listed in the Server Guide 366.

The operation of the Connection Manager 383 is described in greater detail with reference to Figs. 7-11. Turning to Fig. 7, a user navigating the World Wide Web ("WWW" or "the Web") or other interactive content delivery system browses pages containing links to content. For example, a user navigates to a page containing links to the desired content, which is loaded and viewed using a web browser or other viewer capable of rendering pages encoded in Hypertext Markup Language (HTML), step 102. Other navigation and rendering systems are also contemplated by the invention, such as systems based on Gopher or that server pages encoded using alternative markup languages.

Independent of the navigation and rendering system used, the user selects a link to the desired content for playback on the client device, step 104. A check is performed on the client device to determine whether the client has an appropriate plug-in or other software add-on that provides functionality to play back the selected content, step 106. If the necessary plug-in is not present on the client device, it will be retrieved from an available location, step 108. Preferably, the link selected by the user to retrieve the content contains

parameters that instruct the client as to the location of a server containing the necessary plug-in. Alternatively, supplemental links can be provided linking the page containing the link to the content to a server hosting the plug-in required to playback the selected content.

The client determines that the required plug-in is present on the client device and the Connection Manager initiates a connection with and downloads a Server Guide from the Guide Server, step 110. Parameters are provided within the link to the selected content instructing the client where the Guide Server for the selected content is located. Alternatively, a plurality of Guide Servers may be provided to the client whereby the client determines the appropriate Guide Server to initiate a connection with. Furthermore, there is no limitation preventing the Guide Server from being the same server hosting the selected content, e.g., the Content or Media Server. The Server Guide is transmitted from the Guide Server to the client using standard HTTP (Hypertext Transmission Protocol) techniques well known to those skilled in the art or any other suitable data transmission techniques.

The client receives the Server Guide transmitted from the Guide Server via a network and examines the Guide's first entry, step 112. In one embodiment, the Server Guide is a listing of all Content or Media Servers on the network capable of serving the content selected by the user. The Media Servers are preferably listed in order of priority of connection. Alternatively, the Guide Server may store a number of Server Guides, each listing different Media Servers, or listing the same Media Servers in different orders. This alternative allows the Guide Server to select one of the Server Guides based on the current use of resources across all Content Servers, in order to effectuate load balancing.

The Connection Manager is initialized with the address of the supplied server at the top of the Server Guide, step 114. A connection attempt is initiated between the client and the server whereby the Connection Manager tries to establish an acceptable connection with the server, step 116. When a connection is established between the client and the server, the client downloads a Table of Contents. The Table of Contents lists the resources needed to view the content being delivered and channels associated with these resources. The client can then download any missing resources via an appropriate channel. The server sends and received packets to and from the channel it is associated with and maintains statistics, such as numbers of bytes received, number of packets dropped, etc. It also actively monitors and alters bandwidth dynamically. If the client fails to acquire a connection with the Content Server, step 116, the Connection Manager is initialized with a subsequent server address from

the Server Guide, step 118, at which point the Connection Manager once again attempts to initiate a connection with the subsequent server.

Once an acceptable connection is acquired, data is transmitted between the client and Content Server over the communication channel 120. Using techniques described further below, the Connection Manager records data transfer statistics and dynamically alters bandwidth to conform to the transmission requirements of the content for the duration of the transmission, step 122. When the transmission is complete, the communication channel is closed and the routine ends, step 124.

Fig. 8 presents the process involved in initiating and acquiring a connection with a Content Server. The Connection Manager attempts to initiate a connection with the selected Content Server by subscribing to the multicast address which on which data is broadcast from the Content Server, step 130. Multicast is a method for broadcasting data simultaneously to multiple clients across a computer network without maintaining a connection with each client. A check is made to determine if the client successfully subscribed to the multicast address and was able to receive data, step 132. Where the connection is successful, the client will continue to receive multicast data transmitted from the Content Server for the duration of the transmission, step 134.

Fig. 9 presents a block diagram of a client connecting to a Content Server via a multicast router, as described in the preceding paragraph. A multicast client 336a contains and executes Connection Management software 383. The Connection Management software subscribes to a multicast address 372 over network 368 to receive multicast transmissions from content server 160.

Turning back to Fig. 8, if the Connection Manager fails to initiate a connection with a multicast router broadcasting the selected content data, step 132, the Connection Manager references the Server Guide and attempts to initiate a connection with the Content Server via a Multicast-in unicast-out proxy, step 136. A Multicast-in unicast-out proxy is a server that can directly receive a multicast feed while in turn providing a unicast connection with the client. The proxy essentially emulates the multicast router by forwarding the multicast packets over a unicast connection. Each unicast client makes a connection with the Content Server via the multicast-in unicast-out proxy, which is connected via the multicast router. If the client succeeds in making a connection with the Content Server, step 138, unicast UDP data continues to be transmitted to the client via the proxy, step 140.

Referring to Fig. 10, a client is provided that is incapable of receiving multicast packets 336b. The client, through the use of its Connection Management software 383, attempts to make a connection with the multicast address 372 via the data network 368 to receive content from the Content Server 160. When the connection attempt fails, the Connection Manager 383 attempts to access the content via a Multicast-in unicast-out proxy 370. A unicast UDP connection is initiated with the proxy 370, which receives data packets retransmitted by the Multicast Router and forwards them to the client across the unicast connection. This allows the unicast UDP client to receive the transmitted multicast content.

Turning once again to Fig. 8, a client that fails to make a connection via a Multicast-in unicast-out proxy, step 138, references the Server Guide and attempts to make a connection by accessing a Multicast-in unicast-TCP-out proxy, step 142. A Multicast-in unicast-TCP-out proxy Server is a system that responds to TCP based requests for data. Requests generated by the client are posted to the Multicast-in unicast-TCP-out proxy. The Proxy, in turn, maintains a subscription with the multicast router broadcasting the selected content. Packets broadcast by the Content Server to the Multicast Router are received by the Proxy and passed on the client as TCP packets across the unicast TCP connection. If the Connection Manager fails to achieve a connection, step 144, the subroutine ends, step 148.

Fig. 11 presents a configuration of the present invention utilizing a Multicast-in unicast-TCP-out proxy as described in the preceding paragraph. As presented in the previous illustrations, a content server 160 transmits content data to a multicast address 372 to subscribing clients. The client 336c in this situation, unable to receive both multicast and UDP data packets for one of any number of reasons, can only accept TCP packets across a unicast connection. The client 336c initiates a connection with a Multicast-in unicast-TCP-out proxy 374. The Multicast-in unicast-TCP-out proxy 374 receives data broadcast by the multicast router 372. The Multicast-in unicast-TCP-out proxy forwards the received UDP packets as TCP packets across its unicast connection with the client 336c.

Returning again to Fig. 3, producers of multimedia content use the Show Graph Authoring Tool 358 to arrange representations of software elements and their interconnections. The Show Graph Authoring Tool 358 is a standalone software application employing a graphical user interface (GUI) that allows the producer to visually configure representations of software elements that act on data. Each grouping of elements is referred to as a scene and may exist as a subset of other scenes collectively referred to as a Show

Graph 342 or simply a Show. Fig. 12 presents one exemplary series of elements arranged using the Show Graph Authoring tool 391. Elements are visually manipulated on the display with the Producer selecting the desired elements. In its most basic form, the show must consist of a "generator" or element that produces data and a "consumer", the element that manipulates or otherwise acts on the data. This illustration provides as elements an audio-in source generator 388 and an audio-out consumer 392, along with a reverb element 390 that will add a reverb effect to any audio data that it receives.

The Show Graph 342 is an acyclic directed graph, meaning that data moves from one element to the next and will not cycle back from where it came in the graph. Elements comprising a Show Graph 342 are interconnected by signals 394 (Fig. 13) representing pathways that direct data from one element to the next. The elements and their interconnections determine how data is processed and presented to the viewer by the client 336. Fig. 13 builds on the illustration presented in Fig. 12 by illustrating interconnections made by the producer from the audio-in source generator 388 to the reverb 390 and on to the audio-out consumer 392 elements. This show graph 342 provides instructions for the generation of audio, the passing of the resultant data to a reverb element for the addition of reverb, and the sending of the modified audio data to an audio-out for presentation to the viewer.

The Show Graph Authoring Tool 358 retrieves data indicating the current use of resources for a specific client bandwidth, graphics, and CPU capacity. Bandwidth may be measured as the time it takes a client to download a given piece of data. Graphics capacity may be measured as the number of triangles a client can draw to a display device per second. CPU power may be measured as the number of vertices a system can transform per second. With these measurements, the producer derives a "minimum show setting" at or above which the client must function to properly display the show. The producer then defines varying levels at which the show is rendered to provide for both clients with limited and superior bandwidth and CPU capacity. Bandwidth is determined when the taps are laid into the graph.

The Show Graph Authoring Tool 358 further provides the Producer with flexibility as to how to view a scene. The producer can view a show along an animated timeline that sets the elements in order of their timing in the show. Alternatively, the producer can view the show graph as a two dimensional representation of one possible client configuration, or view multiple configurations simultaneously. Client configurations and the

arrangement of elements within a show graph can be freely modified in any mode. Elements and resources in show graphs are identified by globally unique identifiers to support their interchangeability on a system-wide basis. Show graphs 342 are generated and stored on a persistent storage or other type of memory device for transmission to clients 336 requesting a show.

The Show Graph 342 organizes the software components that manipulate Resource data to present a show to a viewer. Resources include, but are not limited to, data such as models used by the Renderers 384, texture maps used by these models, motion data, and audio data.

Client systems vary widely in terms of their computational and graphics power. When a producer generates a Show Graph 342, multiple sets of resources are also created to support show requiring varying amounts of bandwidth and processor power. By creating multiple versions of the same Resources 362 at varying quality levels, the Producer is given the ability to deliver his or her story across a wide spectrum of client systems 336 while maximizing the quality of the presentation of the story. As will be explained herein, the client 336 selects the Resource or Resources that will present the highest quality show given the limitations of the client 336. For example, suppose a Producer generates three 3D models of an actor, one at a resolution of 1000 triangles, another at a resolution of 10,000 triangles and a third at a resolution of 20,000 triangles. Alternatively, Resource resolution is measured in pixels or any other suitable measurement for assessing the resolution of an image. A Client selects the model that will produce the best show possible based upon its specific hardware and bandwidth constraints.

The Media Server 338 also stores and delivers a Table of Contents 344 along with the Show Graph 342 to requesting clients 336. The Table of Contents 344 is a listing of various versions of the Resources 362 required by a Show 342 and the channels on which those Resources are found. As explained above, a channel is an abstraction encompassing a Server 338 or 364 address and a port number where the associated Resource 362 can be found. Each Resource 362 is listed in the Table of Contents 344 in the order of importance of the Resource. The importance of each Resource 362 is equal to the relative value that each Resource has in the delivery of the Show Graph 342. For example, if the story presented to the viewer concerns a news anchor delivering the daily headlines, a high polygon count version of the anchor is more important than a high polygon count of the anchor's desk.

Thus, the listing for the high polygon count version of the anchor would appear in the Table of Contents before a high polygon count version of the desk.

Providing pointers or links to a plurality of identical Resources of varying resolutions or detail allows the system to provide for potentially unlimited scalability. The client 336 determines its capabilities, and therefore the Resources it will retrieve to present a show, through the user of a Benchmarker 382. The Benchmarker measures the client's CPU and graphics power. In one embodiment the Benchmarker 382 collects data regarding CPU time and graphics fill rate. CPU time is defined as the amount of time the CPU spends processing the transformation of vertices in a 3D Renderer 384. The graphics fill rate is a measurement of how much time the 3D Renderer 384 takes to fill triangles and load texture maps. The Renderer 384 calculates execution time by examining a system clock (not pictured), typically part of a general purpose computer's standard hardware.

The Benchmarker 382 performs these calculations several times in succession to derive a weighted average. Averaging helps prevent spurious changes in the data from creating spurious changes in the quality of the Show. It is also used to compensate for irregularities encountered in multithreaded systems. In multithreaded systems, multiple processes are run simultaneously by swapping out each process in a sequential manner. "Swapping out" is a method whereby the state of the processor and the memory used by a process are moved to a section of memory for temporary storage. The processor swaps in the processor state and memory of the next process and runs that process for some predetermined amount of time. This process is repeatedly executed, giving the impression to the user that all processes are running simultaneously.

One side effect of this process is that measurements of the amount of time a process takes using a real world clock may be different than the actual amount of CPU time the process needs. Averaging minimizes these inconsistencies and allows a more accurate appraisal of the capabilities of a client 336. Based on the results of these calculations and Producer set parameters, the client 336 dynamically determines the Resources 362 that provide the best possible viewing experience based on the limitations of the client device 336.

The client 336 contains one or more Renderers or Rendering Engines 384 that receive Resource data 362 and convert it into information the viewer experiences. For example, an audio renderer receives audio resources and produces sound data as output to a speaker 385 integrated or externally connected to the client 336. Likewise, a character

generation renderer takes a stream of text data as input and generates a line or lines of text on a display device 386 integrated or externally connected to the client 336.

As described above, Resources 362 exist at varying levels of complexity. Renderers 384 determine which Resources 362 will be selected from a Table of Contents 344 for a particular show and presented to the viewer. The Benchmarker 382 gives each Renderer 384 the results of its calculations. Based on the calculations, the Renderer 384 modifies the versions of the Resources 362 it selects to conform to these measurements. For example, if a 3D Renderer determines that it can use significantly more CPU time than is currently being used based on the Benchmarker's calculations, it automatically refers to the Table of Contents and selects Resources of a higher resolution. Similarly, if the 3D Renderer has a significantly greater fill rate capacity than is currently being used, more complex and detailed texture map Resources will be located in the Table of Contents and retrieved from the appropriate server.

In preferred embodiments, Media Servers 338 transmit a Show Graph or Graphs 342 to a multicast router 372 for distribution to subscribing clients 366. A Gather Agent 346 analyzes a Show Graph 342 before it is transmitted to requesting clients 336. The Gather Agent 346, and its associated Scatter Agent 380 described in greater detail below, is designed to carry data in a variable capacity buffer and tasked to traverse a Show Graph 342 and do work on the elements therein. When an agent arrives at an element on the Show Graph 342, the number of bytes contained within the element is determined and the Agent expands its buffer to accommodate the data. The agent further issue process calls to each element it analyzes whereby transformation is performed on the data at the element. The process calls manage data coming into and travelling out of an element.

Agents 346 and 380 analyze a Show Graph 342 by traversing all elements contained therein, starting at the Show Node or top level node connecting all other elements in the graph. The Agent 346 and 380 traverses the Show Graph 342 from each Signal 394 destination to its source, seeking a node containing no incoming signal. This type of element is referred to as a "generator" and typically produces digitally encodable input. When a generator is arrived at, the Agent traverses the Show Graph in the reverse direction towards the Show Node. The data collected by the agent is then either delivered to a client or passed to a renderer for presentation to a viewer, depending on whether the agent is a Gather Agent 346 or a Scatter Agent 380.

In accordance with preferred embodiments of the invention, the Gather Agent 346 is a server generated software component. The Gather Agent 346 traverses the Show Graph 342 until a generator is reached. Data is collected from the generator and placed in the Gather Agent 346 buffer. The Gather Agent 346 moves from element to element, following the path between elements defined by the interconnecting signals, and processes the data at each element. In situations where there are multiple paths leading to a Show Node, the Gather Agent 346 traverses each of these child paths. Upon completing traversal of all paths, the Gather Agent 346 has collected all data for the show in its buffer. The buffer is delivered to the client 336 using an operating system specific call. For UNIX and Win32 operating systems, this call is named “sendto”. The sendto call takes three parameters: the data buffer, the destination IP address, and a port number on the destination machine.

A Scatter Agent 380 receives a Gather Agent’s 346 data buffer on this other side of a communication connection. Data is received on the client using an operating specific call referred to as “receivefrom”, which is analogous to the “sendto” command executed on the transmitting machine. The Scatter Agent 380 traverses the Show Graph 342 delivered to the client 336 from a Media Server 338 until the appropriate generator is found. The Agent 380 reverses direction when the appropriate generator is found and moves along Signals 394 from element to element and executes process calls at each element in the path. The Agent 380 arrives at the Show Node where the data in the Agent 380 buffer is acted upon by Renderers 384 for presentation to a viewer.

One embodiment of a process using the system of Figs. 1-3 is shown in Fig. 14. A Media Server generates a Gather Agent in response to a request from a client for a Show, step 396. The Gather Agent identifies the requested Show Graph and begins to traverse the interconnected elements until an element is arrived at with no signal leading into it, e.g., a generator element, step 398. The Gather Agent reverses direction, following Signals from the generator to the next element in the signal path, step 400. If additional nodes are present, step 402, the Gather Agent traverses these elements, step 400, until a Show Node is found indicating no subsequent elements lie along the current signal path. Where additional child paths from the Show Node are present on the Show Graph, step 404, the Gather Agent follows each signal path to its generator and repeats the process of steps 396 through 400.

Once all child paths have been traversed, the Gather Agent’s data buffer is delivered to the client, step 406. Upon receipt of the data buffer from the server, step 408, the

client generates a Scatter Agent initialized with the received data, step 410. Starting at the Show Node, the Scatter Agent traverses the next element on the Show Graph but remains inactive, step 412. The Scatter Agent continues to traverse elements, following signal paths contained in the Show Graph, until a generator is reached, steps 412 and 414. Upon arriving at a generator, the Scatter Agent reverses direction, step 416. The Scatter Agent traverses the subsequent element of the show graph and decodes the data in its buffer intended for the current element by issuing a process call, steps 418 and 420. If the next element is not a Show Node, step 422, the decoding process is repeated, steps 418 and 420. Once the Show Node is arrived at, a check is preformed by the Agent to determine if additional child paths lead from the Show Node to additional generators, step 424. Where additional child paths exist, the process is repeated, steps 412 through 422, to properly decode all data contained within the Scatter Agent. Once all paths have been traversed, the decoded data is passed to the appropriate Renderers to produce output to the viewer, step 426.

Figs. 15-17 present alternative embodiments of the invention whereby a Producer uses the Show Graph Authoring Tool 391 to place Taps 428 and 430 along Signals 394 within a Show Graph that will encode or decode data and allow a Producer to manage the bandwidth of a connection. A Tap 428 and 430 is designed for specific bandwidths. Based on the bandwidth design specified by the Tap, a specific type of encoding is performed. For example, a Tap that needs to satisfy a 28 Kb/sec bandwidth connection would use a compressor that has a high enough compression ratio so that data encoded at the Tap fits into a 28 Kb/sec transmission. Placement of Taps controls which functions within the Show Graph the server handles and which the client handles, thereby establishing a dynamic load balancing. Taps also allow for the use of third party software, such as encryption, and for the development of additional modular programs by third parties which may be more easily brought into a show's development and delivery.

Elements are arranged within the Show Graph Authoring Tool. The Producer places Taps within the Show Graph to determine the most advantageous point to analyze and transmit the data, creating Shows of different quality for clients with different capabilities. Fig. 15 presents a Show Graph containing a low bandwidth tap 428. The Producer looks for points in the Show graph where the data can be encoded and sent to the Client in a form as close to the original as possible. The Producer reviews the Show with the client machine's

capabilities, making judgments regarding acceptable quality. Specific encoding or compression is performed based on the bandwidth design specified by the particular Tap.

The low bandwidth Tap 428 presented in Fig. 15 is placed between the audio in generator 388 and the reverb filter 390. This is the point where the data is voice data and is most easily compressed. The Tap at a low bandwidth encodes data generated by the audio in source 388, e.g., 2.4 Kb/sec, that sounds comparable to the audio produced by a cellular telephone. The Producer sets all Taps in the Show Graph. The taps are set so the sum of bandwidth in all the low bandwidth taps is less than the upper limit on the number of bits that can be reliably received over the client connection.

Fig. 16 presents an alternative embodiment of the Show Graph. Here, a high bandwidth Tap 430 is placed between the reverb filter 390 and the audio out consumer 392. A different high bandwidth Tap 430, such as one based on the mp3 codec, is used. In this illustration, the high bandwidth Tap 430 is placed after the reverb filter 390 so better use can be made of the higher bandwidth. Fig. 17 shows the two Show Graphs presented in Figs. 15 and 16 merged into one Show Graph representing both scenarios. Because the Taps have been defined by specific bandwidth, only clients within their defined bandwidth range act on them.

Placement of Taps 428 and 430 before and after the reverb filter 390 is significant. Filtering involves altering the quality of the data by manipulating input data and transforming it to a different form. By placing the Tap 428 before a filter in the low bandwidth scenario, the data is passed to the client encoded at a low bandwidth without first being reverberated. The encoding entails a sacrifice of quality since the encoder at a low bandwidth will not yield the best sound fidelity. The client receives the data and decodes it using a low bandwidth Tap. The client, through the use of its Scatter Agent, passes the data through a Renderer without assistance from the server since the client is also running a copy of the Show Graph. Decoding on the client side saves crucial bandwidth in transmission and utilizes the computational ability of the client, balancing the processing load between client and server. Conversely, in the high bandwidth Tap scenario the Tap is placed after the reverb filer, allowing the server to perform the reverb conversion and encode the resultant data according to any number of high bandwidth compression methods. The client decodes a voice with reverb at a high sound fidelity, but at the price of high bandwidth consumption. Finally, the use of multiple taps in this fashion allows a client to dynamically modify its

bandwidth and show quality by referencing the appropriate data without active participation by the server.

One embodiment of a process using the system as described is shown in Fig. 18 whereby Taps are placed within a Show Graph and acted upon appropriately by Gather and Scatter Agents. In response to data requested by a client, the server retrieves the requested Show Graph and generates a Gather Agent, step 434. The Gather Agent traverses the Show Graph until arriving at an element without a signal leading into it, e.g., a generator element, and places the data from the element into its buffer, step 436. The Gather Agent traverses subsequent elements comprising the Show graph, step 438, until arriving at a Tap of specified bandwidth, step 440. Each Agent is designed to analyze Taps that have a specific bandwidth. For example, a low bandwidth Agent is designed to analyze a tap encoding and decoding at 22 Kb/sec. This agent would not analyze data at a tap encoding and decoding at 56 Kb/sec.

Once the Agent arrives at the specified Tap, data contained within its buffer is encoded, the Agent deactivates, and traverses the Show Graph until arriving at the Show Node, step 442. A check is performed to determine if additional child paths radiate from the Show Node. Where additional child paths are present, step 444, the process of steps 436 through 442 is performed on each path.

The encoded data buffer is transmitted from the server, step 446, and received by requesting clients, step 448. The client generates a Scatter Agent and initializes it with the received data buffer, step 450. Starting at the Show Node, the Scatter Agent traverses each element in the Show Graph, step 452, until it reaches a generator element, step 454. The Agent reverses direction when it arrives at a generator, but remains inactive, step 456.. The Agent moves away from the generator toward the Show Node, step 460, until it arrives at a Tap where data is delivered to the signal data buffer and decoded, step 462. For example, a client utilizing a low bandwidth Scatter Agent only activates at low bandwidth Taps.

Data in the Scatter Agent's buffer is decoded by the Tap, step 464, after which the Agent processes the decoded data and proceeds along the Show graph executing process calls at each element encountered between the Tap and the Show Node. When the Scatter Agent arrives at the Show Node, a check is made to determine if additional child signal paths radiate from the Show Node, step 466. If additional child paths are present, the Scatter Agent traverses each child path according to steps 452 through 464. Data contained in the Agent's

buffer is decoded and passed to appropriate Renderers to produce output to the viewer, step 468.

Resources 362 are used by the Show Graph 340 to present a show to a viewer. As described above, a Media Server 338 provides a client 336 with both a Show Graph 340 and its associated Table of Contents 344. The client 336, using its benchmarker routine 382, determines the current processing power of the client. These metrics are used as parameters by the Renderers 384 to determine the highest quality Resources 362 listed in the Table of Contents that the client 336 is capable of presenting to the viewer. The client 336 also reads the Table of Contents 344 and examines its own memory to determine which resources of the show are already resident on the client 336.

Fig. 19 presents one embodiment of a process utilizing the system for calculating a client's benchmark metrics and using the calculated data to render the most detailed resources possible. The Benchmarker examines the system clock, step 470. The Benchmarker performs a vertex transformation using a 3D Renderer, step 472. When the vertex transformation is complete, the system clock is again examined to determine how long the transformation took, step 474. This clock measurement is also used to begin timing of a graphics fill rate test whereby the 3D Renderer performs a graphics fill on a set of triangles and loads a series of texture maps, step 476. The system clock is again examined to determine the length of time taken by the calculation, step 478. The Benchmarker receives the results from the 3D Renderer, step 480.

The Benchmarker calculates final CPU processing and graphics fill time by performing a weighted averaging of transformation and fill times, step 482. Averaging allows the Benchmarker to prevent spurious changes in this data from creating spurious changes in the quality of the show presented to the viewer, especially in multithreaded systems. The Benchmarker supplies each Renderer with the current system measurements, step 484, so each may independently determine the proper Resources to utilize given current client performance. Each Renderer receives the benchmark data and, by analyzing the listing of Resources contained in the Table of Contents, retrieves the highest quality versions of Resources capable of currently being supported by the client, step 486. The process is repeated starting at step 470 as client performance is in a constant state of flux.

In order to achieve the desired results of distributing processing loads between servers and clients involved in distribution of rich media, additional steps may be taken

during production of the media to more fully prepare for its distribution. As explained in greater detail below, a producer can divide a media presentation such as video into three components - a 3D virtual set upon which action is to be seen to occur, a recorded or generated video of a person or other actor acting, and data representing the positions of the 2D camera in relation to the actor during the video recording or generation. This positional data, captured in one embodiment by placing IR markers on the 2D cameras and tracking their motion during filming with IR cameras placed at known, fixed positions, is translated to the 3D virtual cameras which relate to the 3D virtual set. Thus, the motion of 2D cameras relative to the actor is used to change the orientation of the virtual set while the 2D image is inserted therein at the client.

Referring to Fig. 21, a system 30 of one preferred embodiment of this aspect of the present invention is implemented in a computer network environment 32 such as the Internet, an intranet or other closed or organizational network. A number of clients 34 and servers 36 are connectable to the network 32 by various means, including those discussed above. For example, if the network 32 is the Internet, the servers 36 may be web servers which receive requests for data from clients 34 via HTTP, retrieve the requested data, and deliver them to the client 34 over the network 32. The transfer may be through TCP or UDP, and data transmitted from the server may be unicast to requesting clients or available for multicasting to multiple clients at once through a multicast router.

In accordance with this aspect of the invention, the server 36 contains several components or systems including a virtual set generator 38, a virtual set database 40, a video processor and compressor 42, and a positional data calculator 44. These components may be comprised of hardware and software elements, or may be implemented as software programs residing and executing on a general purpose computer and which cause the computer to perform the functions described in greater detail below.

Producers of multimedia content use the virtual set generator 38 to develop a three-dimensional model of a set. The model may be based on recorded video of an actual set or may be generated completely based upon computer generated graphical objects. In some embodiments, the virtual set generator includes a 3D renderer. 3D Rendering is a process known to those of skill in the art of taking mathematical representations of a 3D world and creating 2D imagery from these representations. This mapping from 3D to 2D is done in an analogous way to the operation of a camera. The 3D renderer maintains data about the

objects of a 3D world in 3D space, and also maintains the position of a camera in this 3D space. In the 3D renderer, the process of mapping the 3D world onto a 2D image is achieved using matrix mathematics, numerical transforms that determine where on a 2D plane a point in 3D space would project. Meshes of triangles in 3D space represent the surface of objects in the 3D world. Using the matrices, each vertex of each triangle is mapped onto the 2D plane. Triangles that do not fall onto the visible part of this plane are ignored and triangles which fall partially onto this plane are cropped.

The 3D renderer determines the colors for the 2D image using a shader that determines how the pixels for each triangle fall onto the image. The shader does this by referencing a material that is assigned by the producer of the 3D world. The material is a set of parameters that govern how pixels in a polygon are rendered, such as properties about how this triangle should be colored. Some objects may have simple flat colors, others may reflect elements in the environment, and still others may have complex imagery on them. Rendering complex imagery is referred to as texture mapping, in which a material is defined with two traits - one trait being a texture map image and the other a formula that provides a mapping from that image onto an object. When a triangle using a texture mapped material is rendered, the color of each pixel in each triangle is determined by the formulaically mapped pixel in the texture map image.

Virtual sets generated by the set generator are stored in the virtual set database 40 on the server 36, so they may be accessed and downloaded by clients. Models of virtual sets may be considered persistent data, to the extent they do not change over time but rather remain the same from frame to frame of a video show. As a result, models of virtual sets are preferably downloaded from the server 36 to client 34 in advance of transmission of a given video to be inserted in the set. This reduced the bandwidth load required during transmission of the given video data.

The video processor and compressor 42 receives video data 22 recorded by a producer's cameras or generated by a producer through computer animation techniques known to those of skill in the art. In accordance with processes described in greater detail below, the video processor and compressor 42 performs a matting operation on the video to identify separate useful imagery in the video data from non-useful imagery, the useful imagery being that which contains the recorded or generated activity. The video processor 42 further reduces the video to a smaller size by eliminating all or part of the non-useful

imagery, thus compressing it and reduced the bandwidth required for transmission of the video data.

The positional data calculator 44 receives position data 24 recorded or generated by the producer. The position data 24 relates the position the real or virtual camera to the actors in the active portion of the video data 22. As used herein, the term actor is intended to include any object such as a person, animal or inanimate object, which is moving or otherwise changing in the active portion of the video data 22. The positional calculator 44 uses the raw position data 24 to calculate the orientation of the camera with respect to the actor. The client uses this data to position and orient the 3D camera within the virtual set.

The compressed video data and calculated positional data is synchronized and transmitted by the server 36 to any client 34 requesting the data. The client 34 has memory device(s) for storing any virtual sets 48 concurrently or previously downloaded from the server 36, for buffering the video data 50 being received, and for storing the positional data 52. The client contains a video renderer and texture mapper 54, which may be comprised of hardware and/or software elements, which renders the video data within the corresponding virtual set at a location predefined for the virtual set and at a size and orientation as determined based upon the positional data. For example, the orientation of the camera relative to the actor is used to determine the viewpoint to which the three-dimensional model of the virtual set is rotated before rendering as a two-dimensional image. The resulting rendered video and virtual set, and any accompanying audio and other associated and synchronized media signals, is presented on a display 26 attached to the client 34.

One embodiment of a process using the system of Fig. 21 is shown in Fig. 22 and further illustrated in Fig. 23. The virtual set is generated by a producer using 3D modeling tools, step 62, and the completed virtual set is transmitted to a client device for storage, step 64. The set and other imagery in which the talent is placed can be downloaded ahead of time and not re-transmitted with every frame of video. Its texture map imagery is maintained in a known location in memory on the client. Any conventional 3D modeling tool may be used to generate the set, and the virtual set may be, for example, a 3D wireframe model or collection of object models with an image of the set mapped to it. A sample virtual set 92 is shown in Fig. 23 with reference to a virtual camera 93 that indicates the viewpoint from which the set may be viewed.

Infrared markers are placed within the talent space, step 66, e.g., on the cameras filming the talent and possibly at other locations. Talent is video recorded on a blue background, step 68, and the camera positional data is captured, step 72. Referring also to Fig. 4, by placing talent 94 on a blue background 95, the video of the talent recorded by a camera 16 can be sent to a chroma keyer 96, a stand alone piece of hardware on the server side of the connection. The chroma keyer generates high contrast black and white imagery 97, step 74 (Fig. 22), in which the talent 94 appears as a white stencil on a black background. A combiner/encoder 98 uses a video compression algorithm to recombine the video of the talent over the blue screen, and the output of the chroma keyer, step 76. The system thus detects where the talent is and is not. This consequently removes the need to encode black image data on the screen. The image is cropped down to a rectangle or other polygon comprising the white image of the talent, step 78, and the black imagery remaining inside the rectangle is transparent, step 80.

Only the rectangle the talent occupies is compressed and transmitted to the client, step 82, along with the positional data, step 84. Because the amount of video and other data transmitted is small, and the amount of data needed to represent the camera is small, transmission of the virtual set such as over the Internet takes better advantage of low bandwidth than existing video compression technologies. In some embodiments, the video portion of talent on a set is a small percentage of the total raster, typically 10-25%. With the smaller image, extra data space can be used to increase frame time or increase the resolution of the imagery or for the insertion of advertising.

The Client uses the compressed video as input into a texture map. A texture mapper is a 3D rendering tool that allows a polygon to have a 2D image adhered to it. The texture map's imagery is comprised of the transmitted video and subsequent changes on a frame-to-frame basis. The client decompresses the video and places it in the known location within the virtual set, step 86. This image can comprise both color and transparency. Where there is blue screen the texture map is transparent. Where there is no blue the pixels of the talent appear. This rendered image gives the impression that the talent is in the virtual set.

The client uses the virtual set camera position to position the 3D renderer's camera and manipulate the virtual set, step 88. By matching the 3D camera's position to the real camera's position, the video retains its dimensionality. By tracking the real camera on

the blue set and transferring this data to the 3D camera in the 3D virtual set, real motion on the real set becomes virtual motion on the virtual set.

As explained above, the position of the camera within the blue set is tracked by placing infrared markers at strategic positions on the camera. Infrared sensitive cameras positioned at known stationary points in the blue set detect these markers. The position of these markers in 3D space in the blue set is detected by triangulation. Fig. 24 is a top down view of two 2D cameras 16 taking the position of an infrared marker 99. Both cameras 16 have unique views represented by the straight lines vectoring from the cameras in Fig. 24. These lines indicate the plane on which the real world is projected in the camera. Both cameras are at known positions. The circles 99' on the fields of view represent the different points at which the infrared marker 99 appears on the cameras. These points are recorded and used to triangulate the position of the marker in 3D space, as known to those of skill in the art.

Because a virtual set tells which part of the screen is useful, the amount of bandwidth required to deliver each frame to the client is greatly reduced. The processing and compression of the video data as described herein reduces the video data transmitted to the client from full raster, full video screen, edge to edge, top to bottom, to only the amount where the action is taking place. Only a small portion of the raster has to be digitized. In addition, because the persistent data with regard to the show is pre-transmitted and already resides on the client, the system and method of the present invention are able to do more at a larger screen size with a higher resolution image than conventional compressed/streaming video are able to achieve.

The virtual set system of the present invention may be utilized with the media engine described above, to maximize the opportunities to distribute processing load between servers and clients during video distribution over a network.

While the invention has been described and illustrated in connection with preferred embodiments, many variations and modifications as will be evident to those skilled in this art may be made without departing from the spirit and scope of the invention, and the invention is thus not limited to the precise details of methodology or construction set forth above as such variations and modifications are intended to be included within the scope of the invention.

WHAT IS CLAIMED IS:

1. A method for preparing a multimedia presentation for transmission to a client over a network, the method comprising:

allowing a producer of the presentation to identify elements of software which process data representing resources used in the presentation;

allowing the producer to specify connections between two or more elements, wherein a connection represents a flow of data from one element to another element;

allowing the producer to specify a plurality of resources to be processed by the identified elements and location data indicating the locations of the resources on one or more servers connectable to the network; and

generating a set of presentation data structures, including the identified elements, connections, and resources, for transmission to a client to thereby enable the client to reproduce the multimedia presentation from the presentation data structures by retrieving at least some of the resources and processing the retrieved resources with the identified elements in accordance with the specified connections.

2. The method of claim 1, wherein generating the set of presentation data structures comprises generating a show graph representing a plurality of identified elements and the connections extending therebetween.

3. The method of claim 1, wherein generating the set of presentation data structures comprises generating a table listing all or some of the specified resources and corresponding locations.

4. The method of claim 1, comprising allowing the producer to replace a first identified element with a second element while retaining for the second element any connection and resources associated with the first element.

5. The method of claim 1, comprising allowing the producer to specify an encoder to tap a signal on a selected connection and encode the data flowing at the selected connection.

6. The method of claim 5, wherein allowing the producer to specify the encoder comprises allowing the producer to associate the specified encoder with a given set of client processing capabilities or available bandwidth for transmission to a client.

7. The method of claim 6, comprising:

receiving data indicating the processing capabilities of a client requesting the presentation or available bandwidth for transmission of the presentation to the client; and

traversing a series of identified elements via their connections until a tapped encoder is located being associated with the client's processing capabilities or available bandwidth.

8. The method of claim 7, comprising encoding the data at the tapped connection using the specified encoder.

9. The method of claim 8, comprising transmitting the encoded data to the requesting client.

10. The method of claim 9, comprising the client:

receiving the presentation data structures;

traversing the series of identified elements via their connections until a tapped encoder is located being associated with the client's processing capabilities, available bandwidth, or both; and

decoding the data at the tapped connection using a decoder corresponding to the specified encoder.

11. A system for delivering a multimedia presentation from a server to a client, the system comprising:

a presentation-authoring tool for use by a producer of the presentation to

identify software elements for processing data representing resources used in the presentation,

specify connections between two or more elements representing a flow of data from one element to another element,

specify a plurality of resources to be processed by the identified elements and location data indicating the locations of the resources on one or more servers connectable to the network, and

specify an encoder to tap a signal on a selected connection and encode the data flowing at the selected connection;

and

a first agent for traversing a series of identified elements via their connections until a tapped encoder is located and encoding the data at the tapped connection using the specified encoder.

12. The system of claim 11, wherein the authoring tool allows the producer to associate a specified encoder with a given set of client processing capabilities or available bandwidth for transmission to a client.

13. The system of claim 12, comprising a plurality of first agents, each associated with a different set of client processing capabilities or available bandwidth.

14. The system of claim 13, comprising means for receiving data indicating the processing capabilities or available bandwidth for a client requesting the presentation and selecting one of the first agents associated with the indicated capabilities or bandwidth for traversing the series of identified elements.

15. The system of claim 11, comprising a second agent for traversing a series of identified elements via their connections until a tapped decoder is located and decoding the data at the tapped connection using the specified decoder.

16. A method for customizing a multimedia presentation based upon processing capabilities of a client requesting the presentation or available bandwidth for transmission to the client, the method comprising:

storing data structures identifying a series of software elements for processing a resource representing data, forming part of the presentation and connections between the software elements in the series on which resource data flows from one element to another;

specifying a plurality of encoders to tap various connections within the series of software elements, the encoders each being associated with a given set of client processing capabilities, available bandwidth for transmission to a client, or both; and

based upon the client's processing capabilities, available bandwidth, or both, selecting one of the encoders and encoding the data flowing along the connection tapped by the selected encoder using the selected encoder.

17. A method implemented by a client computer for retrieving a multimedia presentation from a server over a network and presenting the presentation, the method comprising:

performing one or more benchmarking tests on the client computer to determine one or more operational parameters of the client computer;

retrieving a presentation data structure from the server identifying a plurality of software elements and data resources used in reproducing the presentation, the software elements and resources being associated with varying types of operational parameters;

selecting a subset of the elements and resources based upon the client's operational parameters determined in the benchmarking tests; and

retrieving the selected elements and resources to thereby reproduce the presentation.

18. The method of claim 17, wherein performing one or more benchmarking tests comprises testing the client computer's CPU speed.

19. The method of claim 18, wherein testing CPU speed comprises measuring time spent by the CPU processing transformations of vertices in a three-dimensional renderer.

20. The method of claim 17, wherein performing one or more benchmarking tests comprises testing the client computer's graphics fill rate.

21. The method of claim 20, wherein testing the graphics fill rate comprises measuring time spent by a three-dimensional renderer running on the client computer in filling triangles.

22. The method of claim 20, wherein testing the graphics fill rate comprises measuring time spent by a three-dimensional renderer running on the client computer in reading texture maps.

23. A computer readable medium storing program code for, when executed, causing a computer to perform a method for retrieving a multimedia presentation from a server over a network and presenting the presentation, the method comprising:

performing one or more benchmarking tests on the client computer to determine one or more operational parameters of the client computer;

retrieving a presentation data structure from the server identifying a plurality of software elements and data resources used in reproducing the presentation, the software elements and resources being associated with varying types of operational parameters;

selecting a subset of the elements and resources based upon the client's operational parameters determined in the benchmarking tests; and

retrieving the selected elements and resources to thereby reproduce the presentation.

24. The medium of claim 23, wherein the step performed by the computer of performing one or more benchmarking tests comprises testing the client computer's CPU speed.

25. The method of claim 24, wherein the step performed by the computer of testing CPU speed comprises measuring time spent by the CPU processing transformations of vertices in a three-dimensional renderer.

26. The method of claim 23, wherein the step performed by the computer of performing one or more benchmarking tests comprises testing the client computer's graphics fill rate.

27. The method of claim 26, wherein the step performed by the computer of testing the graphics fill rate comprises measuring time spent by a three-dimensional renderer running on the client computer in filling triangles.

28. The method of claim 26, wherein the step performed by the computer of testing the graphics fill rate comprises measuring time spent by a three-dimensional renderer running on the client computer in reading texture maps.

29. A method for distributing video over a network for display on a client device, the method comprising:

storing model data representing a set in which action occurs;

generating video data representing action occurring;

capturing positional data representing a position of a camera during the action in generated video; and

transmitting from a server to the client device as separate data items the model data, generated video, and positional data, to thereby enable the client device to reproduce and display a video comprising the action occurring at certain positions within the set.

30. The method of claim 29, comprising transmitting the model data in advance of the video and positional data.

31. The method of claim 30, comprising the client device persistently storing the transmitted model data for use with a plurality of video and positional data items.

32. The method of claim 29, comprising, prior to transmission to the client, cropping the generated video data to eliminate some or all portions of the video in which no action occurs.

33. The method of claim 32, wherein cropping the generated video data comprises matting the video to separate the action from other portions of the video data.

34. The method of claim 33 wherein matting the video comprises generating a high contrast black and white image of the video wherein a white portion of the image represents the action, and cropping out all or part of a black portion of the image.

35. The method of claim 34, wherein generating a high contrast image comprises processing the video using a chroma keyer.

36. The method of claim 35, wherein generating the video data comprises recording action occurring in front of a blue screen, and wherein generating the high contrast image comprises using a chroma keyer on the recorded video.

37. The method of claim 29, wherein capturing positional data comprises capturing data representing the position of the camera with respect to the action in the video data.

38. A method for receiving video over a network and presenting it on a client device, the method comprising:

receiving from a server as separate data items model data representing a set in which action occurs, video data representing action occurring, and positional data representing the position of the camera during the action in the generated video;

rendering the video data within the set at a position within the set determined using the positional data to thereby produce the video; and

presenting the video on a client device.

39. The method of claim 38, wherein the model data comprises graphical data representing a three-dimensional virtual set.

40. The method of claim 39, wherein the graphical data is configured to be rendered as a two-dimensional image at a plurality of viewing angles relative to a virtual camera.

41. The method of claim 40, wherein the positional data comprises orientation data representing the position of the virtual camera relative to the action in the video data, and wherein rendering the video data within the set comprises selecting a viewing angle for the set using at least the orientation data.

42. The method of claim 39, wherein rendering the video data within the set comprises mapping the video data as a texture map onto the model data.

43. A method for distributing video over a network, the video representing an actor in motion, the set being represented in a three-dimensional rotatable model stored on a client connected to the network, the method comprising:

eliminating all or part of the video not containing the actor including matting the video to separate the actor from other parts of the video;

transmitting from a server to the client as separate data items the video and positional data representing the position of the real camera relative to the actor in the video;

the client receiving the video and positional data;

the client determining based upon the positional data whether to rotate the three-dimensional model of the set to properly orient the video therein, and rotating the model accordingly;

the client rendering the video within the rotated model at a depth determined based upon the positional data; and

the client presenting the rendered video and set.

44. A system for preparing a video for distribution over a network to one or more clients, the video containing one or more actors, the system comprising:

a positional data capturing system for capturing position data representing a position of the one camera relative to the actors in the video;

a video compression system for reducing the video by eliminating all or a portion of the video not containing the actor, the video compression system including a matting system for matting the video to separate the actor from other parts of the video; and

a transmission system for transmitting compressed video in association with corresponding positional data and in association with model data representing a set within which the video is rendered for presentation by one or more clients.

45. A computer implemented method for managing a retrieval of multimedia content over a computerized network, the network having a plurality of servers connectable to one or more clients, the method comprising:

(a) retrieving at a first client a server guide identifying a list of servers capable of delivering a selected item of multimedia content;

(b) the first client automatically determining whether a connection may be made to a first server identified in the server guide to achieve delivery of the selected content item;

(c) if the connection may be made, the first client establishing a connection with the first server to retrieve the selected content item therefrom;

(d) if the connection is unable to be made, the first client automatically determining whether a connection may be made to a second server identified in the server guide to achieve delivery of the selected content item; and

the first client repeating steps (c) and (d) for the second server and any additional server identified in the server guide until a connection may be made to a server by which the selected content item may be delivered.

46. The method of claim 45, comprising selecting the multimedia content item from a list of available items.

47. The method of claim 45, wherein retrieving the server guide comprises retrieving the guide from a guide server, and comprising the guide server storing a plurality of server guides for the content item and selecting one of the stored server guides upon receipt of a request from the client.

48. The method of claim 45, wherein the servers identified in the server guide include one or more routers connectable to a content server, the content server storing the selected content item.

49. The method of claim 48, wherein the first server is a multicast router and the second server is a multicast-in unicast-out proxy configured to receive data from the multicast router and provide a unicast connection to the first client.

50. The method of claim 49, wherein a third server identified in the server guide is a multicast-in unicast-TCP-out proxy configured to receive requests for parts of the content item from clients, subscribe to the multicast router, and deliver to clients data packets representing requested parts of the content item.

51. The method of claim 50, wherein the steps of automatically determining whether a connection may be made are performed first for the multicast address, then for the multicast-in unicast-out proxy router, and then for the multicast-in unicast-TCP-out proxy.

52. The method of claim 45, wherein the server guide lists the servers in a given sequence, and wherein the steps of automatically determining whether a connection may be made are performed according to the given sequence of servers listed in the server guide.

53. The method of claim 45, wherein the server guide identifies each server in the list through a server address and server type.

54. A computer implemented method for managing a retrieval of multimedia content from a content server over a computerized network, the network having a plurality of servers connectable to one or more clients, the method comprising:

retrieving at a first client a server guide identifying a list of servers capable of delivering a selected item of multimedia content from the content server, the list including a multicast router and a multicast-in unicast-out proxy router;

the first client automatically determining whether a connection may be made to the multicast router identified in the server guide to achieve delivery of the selected content item;

if the connection may be made to the multicast router, the first client establishing a connection with the multicast router to retrieve the selected content item therefrom;

if the connection is unable to be made to the multicast router, the first client automatically determining whether a connection may be made to the multicast-in unicast-out proxy router identified in the server guide to achieve delivery of the selected content item; and

if the connection may be made to the multicast-in unicast-out proxy router, the first client establishing a connection with the multicast-in unicast-out proxy router to retrieve the selected content item therefrom.

55. The method of claim 54, wherein the list of servers further includes a multicast-in unicast-TCP-out proxy, and comprising, if the connection is unable to be made to the multicast-in unicast-out proxy router, the first client automatically determining whether a connection may be made to the multicast-in unicast-TCP-out proxy identified in the server guide to achieve delivery of the selected content item.

56. A computer readable medium storing program code for, when executed, causing a computer to perform a method for managing a retrieval of multimedia content over a computerized network, the network having a plurality of servers connectable to one or more clients, the method comprising:

(a) retrieving at a first client a server guide identifying a list of servers capable of delivering a selected item of multimedia content;

(b) the first client automatically determining whether a connection may be made to a first server identified in the server guide to achieve delivery of the selected content item;

(c) if the connection may be made, the first client establishing a connection with the first server to retrieve the selected content item therefrom;

(d) if the connection is unable to be made, the first client automatically determining whether a connection may be made to a second server identified in the server guide to achieve delivery of the selected content item; and

the first client repeating steps (c) and (d) for the second server and any additional server identified in the server guide until a connection may be made to a server by which the selected content item may be delivered.

57. A system for establishing a connection over a network to retrieve multimedia content, the system comprising:

a memory device storing a server guide identifying a list of servers capable of delivering a selected item of multimedia content, the list including servers differing in transmission techniques; and

a connection manager for automatically attempting to establish a connection to the servers contained in the list one at a time and, upon determining that a connection can not be established for a given server, attempting to establish a connection to another server in the list until a connection is established or connections can not be established to all servers.

58. The system of claim 57, wherein the list of servers includes at least one server configured to multicast the content item and at least one server configured to unicast the content item.

59. The system of claim 58, wherein the list of servers includes two or more of the following: a multicast router, a multicast-in unicast-out proxy router, and a multicast-in unicast-TCP-out proxy.

60. A computer implemented method for receiving content data transmitted from a server in a sequence of packets, the server repeatedly transmitting the packets in sequence, the method comprising:

upon receipt of a first packet from the server, deriving from the packet a number of the packet in the sequence and a total number of packets in the sequence;

generating and storing an index having an entry for each of the packets in the sequence based upon the total number of packets in the sequence;

updating the index for each packet received subsequent to the first packet;

detecting based on the index whether any subsequent packet is missing from the sequence of packets; and

if a missing packet is detected, determining whether the first time required to retrieve the missing packet by waiting for the packet to be received in the repeating sequence is greater than a threshold time and, if the first time is greater than the threshold time, requesting the missing packet to be delivered from a server.

61. The method of claim 60, wherein updating the index comprises deriving the packet number from the subsequent packet in the sequence and updating the entry in the index corresponding to the derived packet number.

62. The method of claim 61, wherein detecting whether any subsequent packet is missing comprises comparing the packet number for a currently received subsequent packet to the packet number for the packet received immediately prior to the currently received packet to determine whether the currently received packet number follows consecutively in the sequence from the immediately prior packet number.

63. The method of claim 61, comprising detecting a second receipt of the first packet based on the index.

64. The method of claim 63, comprising stopping further receipts of packets in the sequence upon detection of the second receipt of the first packet.

65. The method of claim 63, wherein detecting whether any subsequent packet is missing comprises checking the index for any missing packet number following detection of the second receipt of the first packet.

66. The method of claim 60, wherein deriving the packet number and total number of packets comprises retrieving the packet number and total number from a header of the packet.

67. The method of claim 60, comprising estimating based upon the first packet a total data size for the packets in the sequence and allocating a storage buffer for the packets in the sequence at least as large as the total data size.

68. The method of claim 67, wherein estimating the total data size comprises determining the size of the first packet and multiplying the first packet size by the total number of packets.

69. The method of claim 67, comprising storing content data in received packets in the allocated storage buffer in a sequence corresponding to the packet numbers.

70. The method of claim 60, wherein the threshold time comprises a time required to request and receive the missing packet from the server.

71. The method of claim 60, wherein determining whether the first time is greater than the threshold time comprises computing the first time based upon data representing available bandwidth.

72. The method of claim 60, wherein determining whether the first time is greater than the threshold time comprises computing the first time based upon measured time for receiving packets in the sequence.

73. The method of claim 60, comprising the client receiving packets by issuing a subscription request to a multicast server.

74. A system for delivering content from a server to one or more clients, the system comprising:

a multicast server for transmitting an item of content in a sequence of packets, each packet containing a header storing a number of the packet in the sequence, the packets being transmitted as repeating loops of the packets in sequence;

a client system for subscribing to the multicast server, receiving the transmitted packets, tracking the receipt of packets using the packet numbers, identifying any packets in the sequence which are missing based on the tracked packet numbers, and deciding whether to wait for any given missing packet to be received in the subsequent loop or request the missing packet from the server;

the multicast server transmitting the missing packet in response to a request received from the client system.

75. The system of claim 74, wherein the header for at least one packet contains data representing a total number of the packets in the sequence.

76. The system of claim 75, wherein the total number of packets is contained in the header for each packet in the sequence.

77. The system of claim 75, wherein the client system comprises a memory device storing an index of packet numbers, the index having a number of entries equal to the total number of packets in the sequence and being used by the client system in tracking .

78. The system of claim 75, wherein the client system comprises a memory device storing packets, the client system allocating space within the memory device for storage of the packets based on the total number of packets and a data size for at least one of the packets.

79. The system of claim 74, wherein the multicast server comprises a packetized data source structure for decomposing the content into the sequence of packets.

80. A computer readable medium storing program code for, when executed, causing a computer to perform a method for receiving content data transmitted from a server in a sequence of packets, the server repeatedly transmitting the packets in sequence, the method comprising:

upon receipt of a first packet from the server, deriving from the packet a number of the packet in the sequence and a total number of packets in the sequence;

generating and storing an index having an entry for each of the packets in the sequence based upon the total number of packets in the sequence;

updating the index for each packet received subsequent to the first packet;

detecting based on the index whether any subsequent packet is missing from the sequence of packets; and

if a missing packet is detected, determining whether the first time required to retrieve the missing packet by waiting for the packet to be received in the repeating sequence is greater than a threshold time and, if the first time is greater than the threshold time, requesting the missing packet to be delivered from a server.

81. A computer implemented method for transmitting content data from a server to one or more clients, the method comprising:

breaking the content data into a plurality of data packets arranged in a sequence;

inserting into a header of each data packet a total number of packets in the sequence, a number of the particular packet in the sequence, and data representing a size of the data in the particular data packet;

repeatedly transmitting the data packets in sequence; and

upon request from a client for transmission of a given data packet, transmitting the given data packet in response to the request without waiting for the packet to appear in sequence.

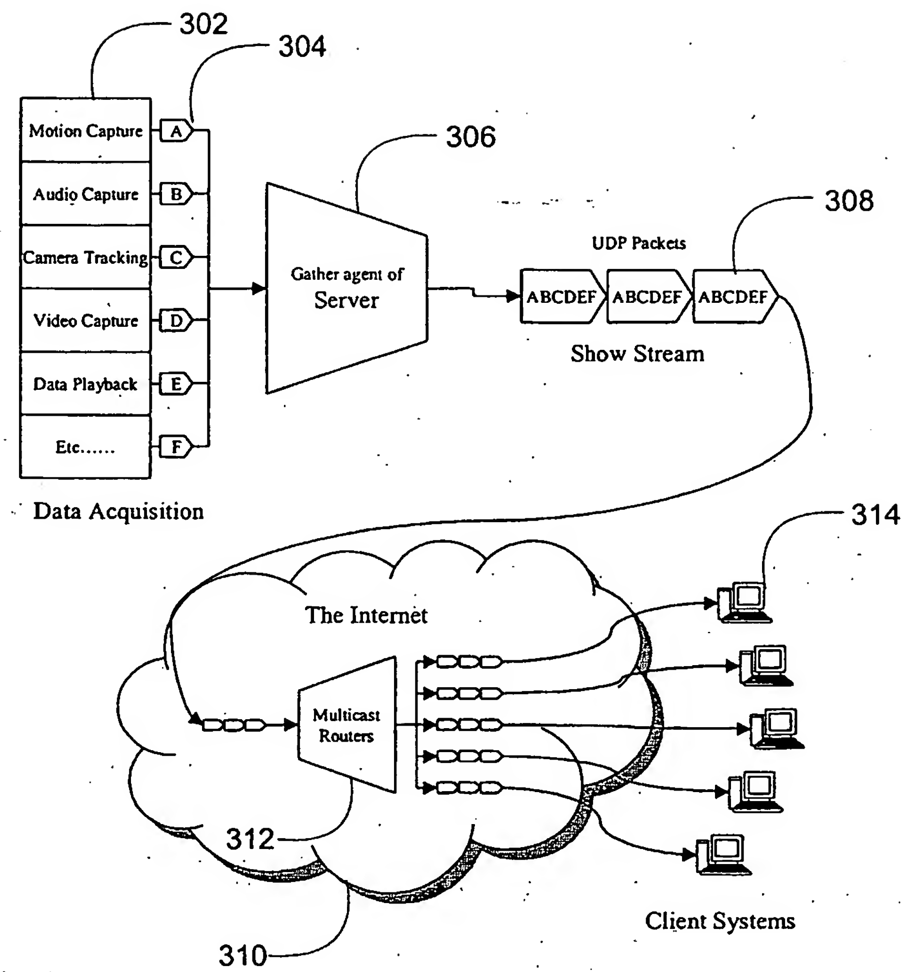

302
304
Motion Capture
Audio Capture
Camera Tracking
Video Capture
Data Playback
Etc......
A
B
C
D
E
F
Data Acquisition
306
Gather agent of
Server
308
UDP Packets
ABCDEF
ABCDEF
ABCDEF
Show Stream
314
The Internet
Multicast
Routers
312
310
Client Systems

Fig. 1

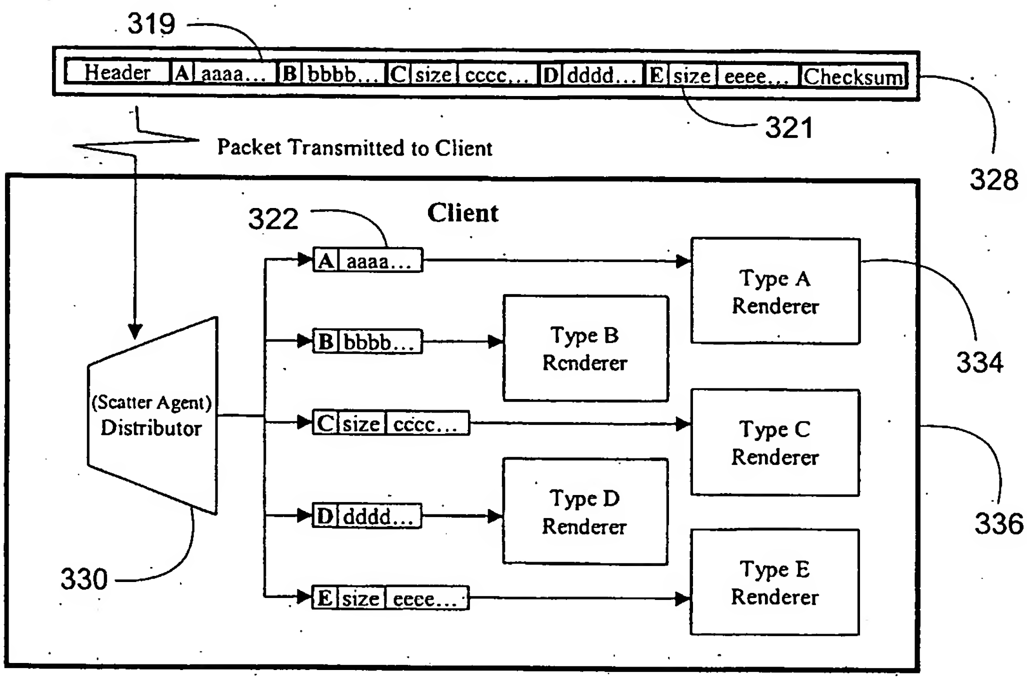

319
Header
A aaaa...
B bbbb...
C size cccc...
D dddd...
E size eeee...
Checksum
321
328
Packet Transmitted to Client
Client
322
A aaaa...
B bbbb...
C size cccc...
D dddd...
E size eeee...
(Scatter Agent) Distributor
Type A Renderer
Type B Renderer
Type C Renderer
Type D Renderer
Type E Renderer
330
334
336

Fig. 2

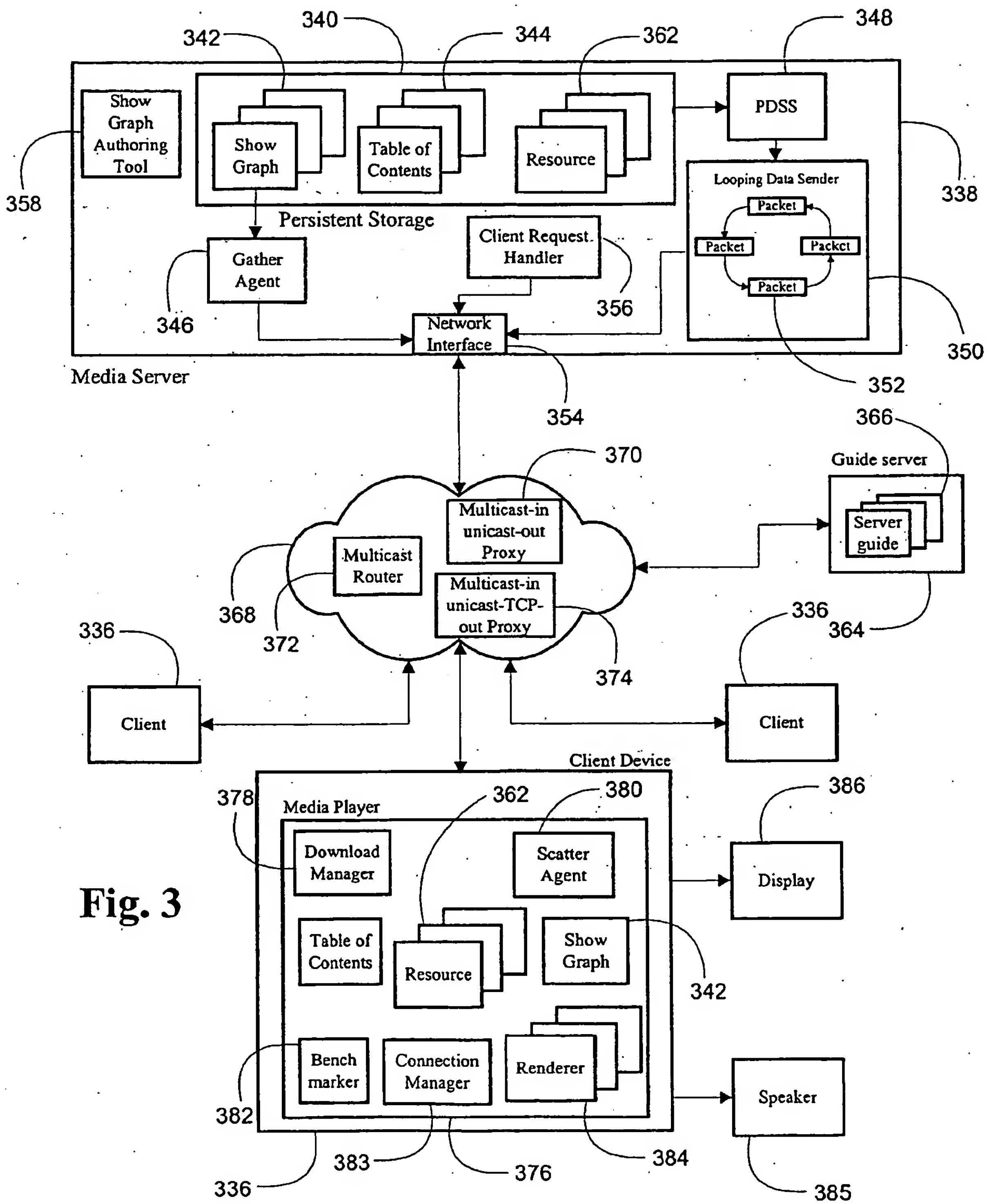

Show Graph Authoring Tool
Show Graph
Table of Contents
Resource
Persistent Storage
PDSS
Looping Data Sender
Packet
Packet
Packet
Packet
Client Request Handler
Gather Agent
Network Interface
Media Server
Guide server
Server guide
Multicast Router
Multicast-in unicast-out Proxy
Multicast-in unicast-TCP-out Proxy
Client
Client
Client Device
Media Player
Download Manager
Scatter Agent
Table of Contents
Resource
Show Graph
Bench marker
Connection Manager
Renderer
Display
Speaker
340
342
344
362
348
358
338
346
356
350
352
354
366
370
368
372
336
374
364
378
380
386
342
382
383
376
384
385

Fig. 3

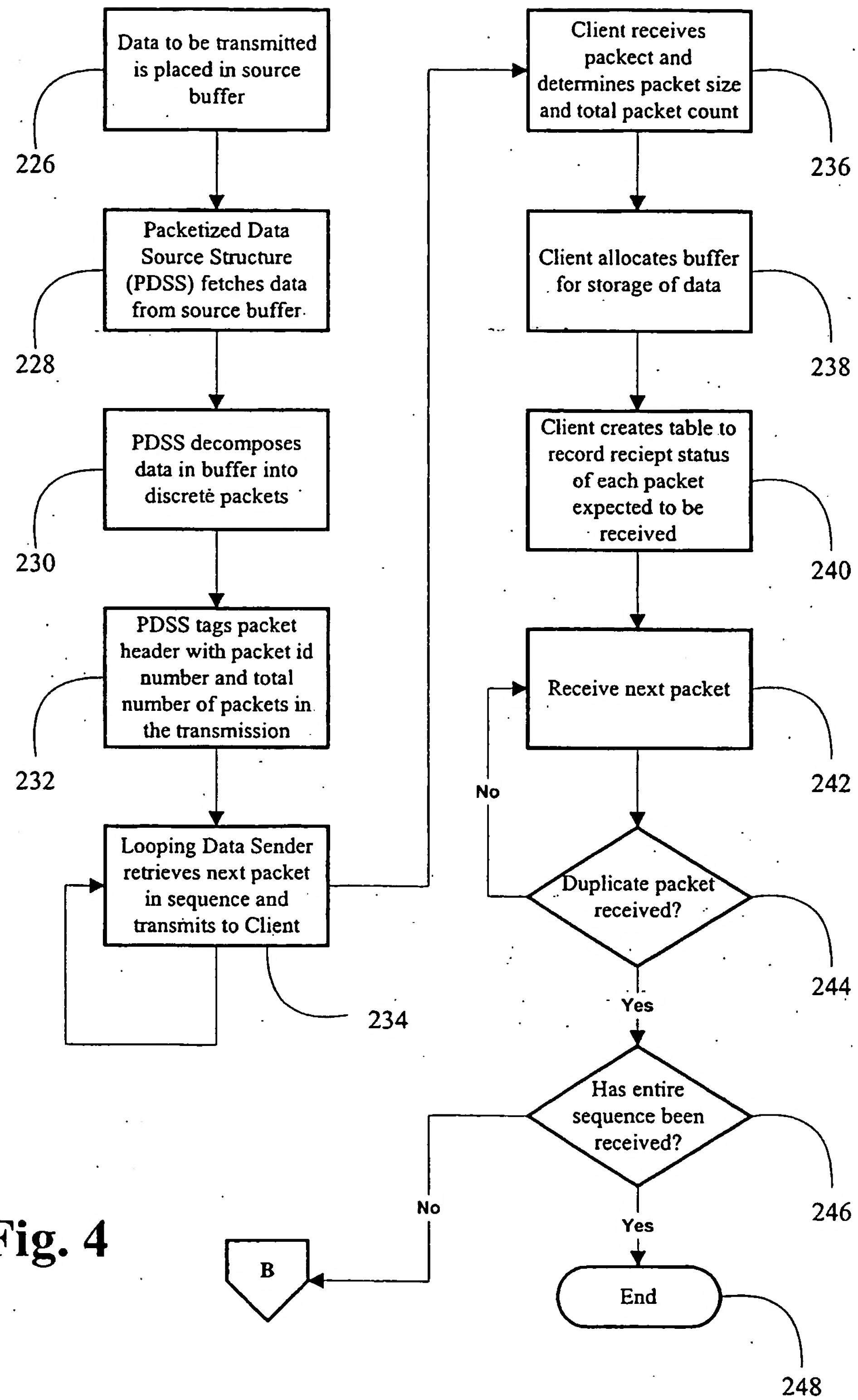

Data to be transmitted is placed in source buffer
226
Packetized Data Source Structure (PDSS) fetches data from source buffer
228
PDSS decomposes data in buffer into discrete packets
230
PDSS tags packet header with packet id number and total number of packets in the transmission
232
Looping Data Sender retrieves next packet in sequence and transmits to Client
234
Client receives packect and determines packet size and total packet count
236
Client allocates buffer for storage of data
238
Client creates table to record reciept status of each packet expected to be received
240
Receive next packet
242
No
Duplicate packet received?
244
Yes
Has entire sequence been received?
246
No
Yes
B
End
248

Fig. 4

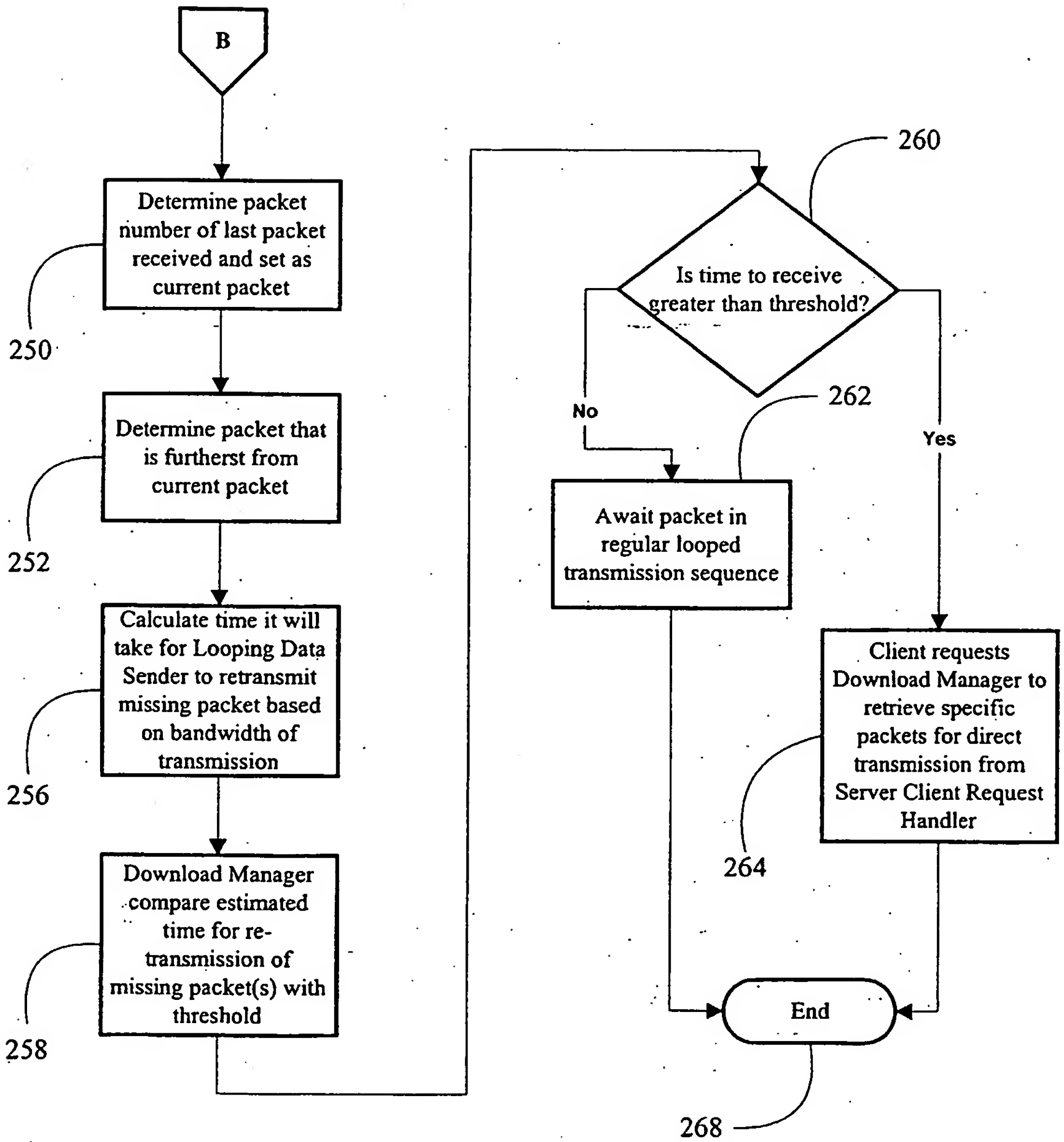

B
Determine packet number of last packet received and set as current packet
250
Determine packet that is furtherst from current packet
252
Calculate time it will take for Looping Data Sender to retransmit missing packet based on bandwidth of transmission
256
Download Manager compare estimated time for re-transmission of missing packet(s) with threshold
258
Is time to receive greater than threshold?
260
No
Yes
262
Await packet in regular looped transmission sequence
Client requests Download Manager to retrieve specific packets for direct transmission from Server Client Request Handler
264
End
268

Fig. 5

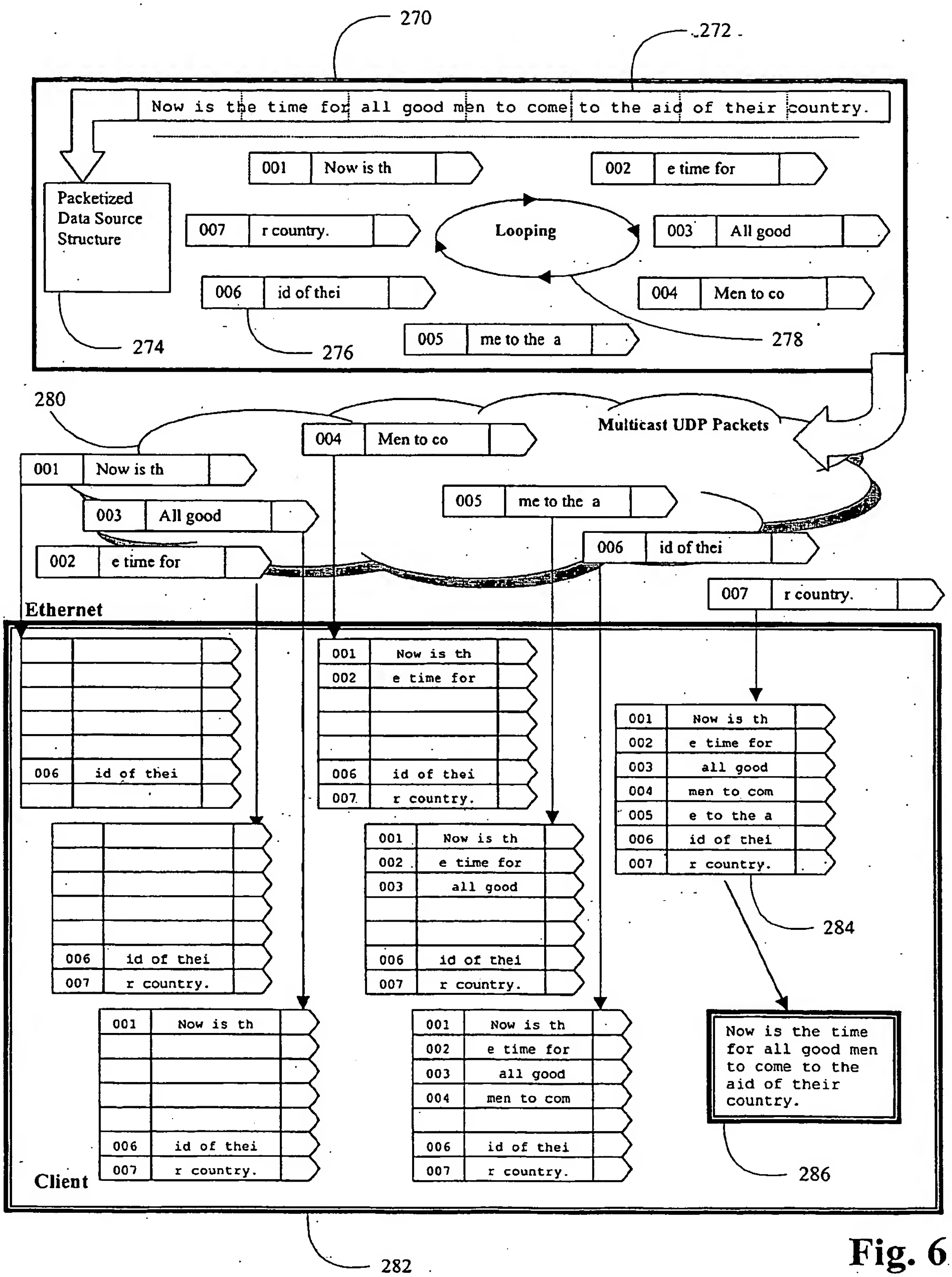

270
272
Now is the time for all good men to come to the aid of their country.
Packetized Data Source Structure
001 Now is th
002 e time for
003 All good
004 Men to co
005 me to the a
006 id of thei
007 r country.
Looping
274
276
278
280
Multicast UDP Packets
Ethernet
Client
284
Now is the time for all good men to come to the aid of their country.
286
282

Fig. 6

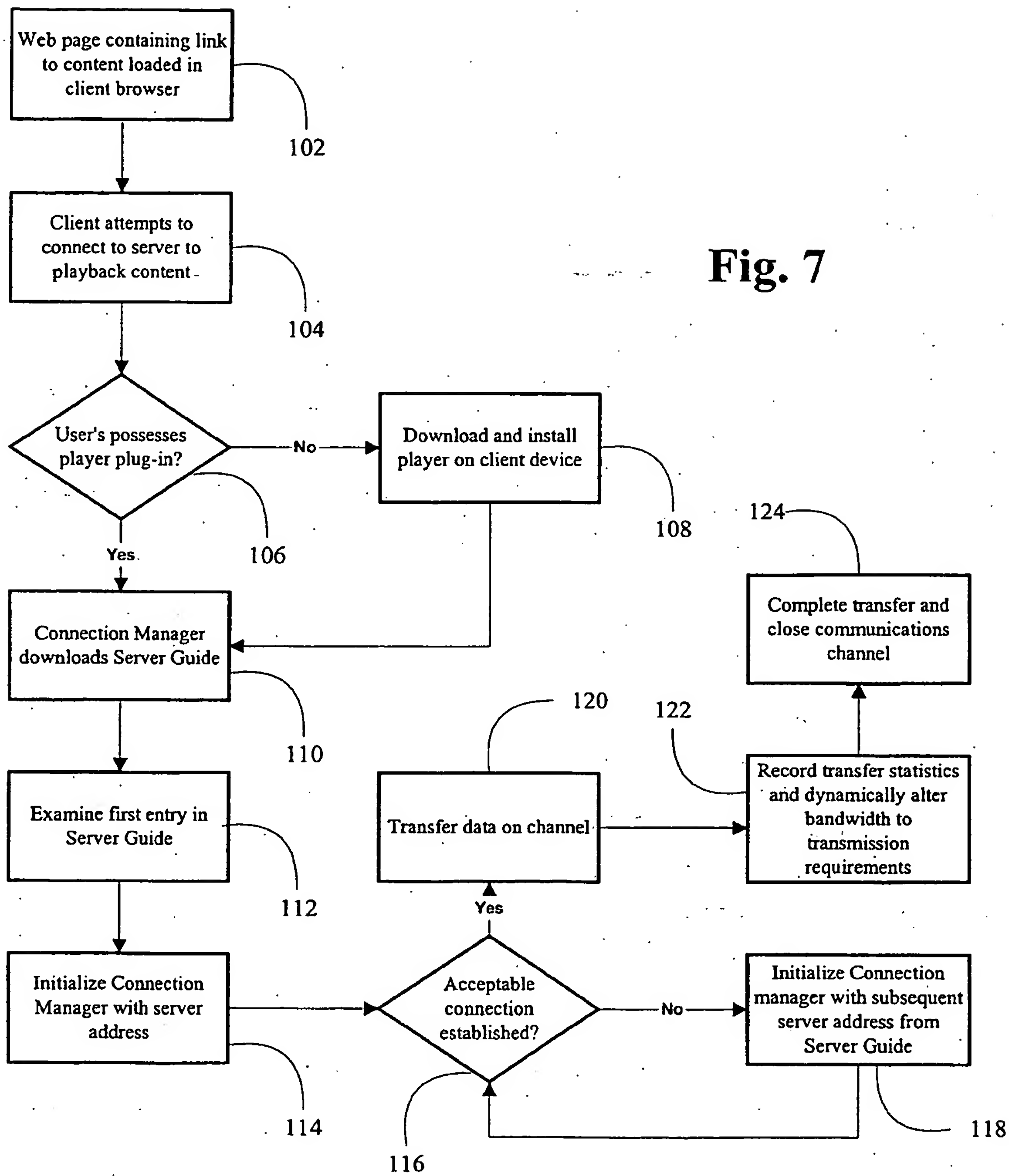

Web page containing link to content loaded in client browser
102
Client attempts to connect to server to playback content
104
User's possesses player plug-in?
No
Download and install player on client device
108
106
Yes
Connection Manager downloads Server Guide
110
Examine first entry in Server Guide
112
Initialize Connection Manager with server address
114
Acceptable connection established?
Yes
No
116
Transfer data on channel
120
Record transfer statistics and dynamically alter bandwidth to transmission requirements
122
Complete transfer and close communications channel
124
Initialize Connection manager with subsequent server address from Server Guide
118

Fig. 7

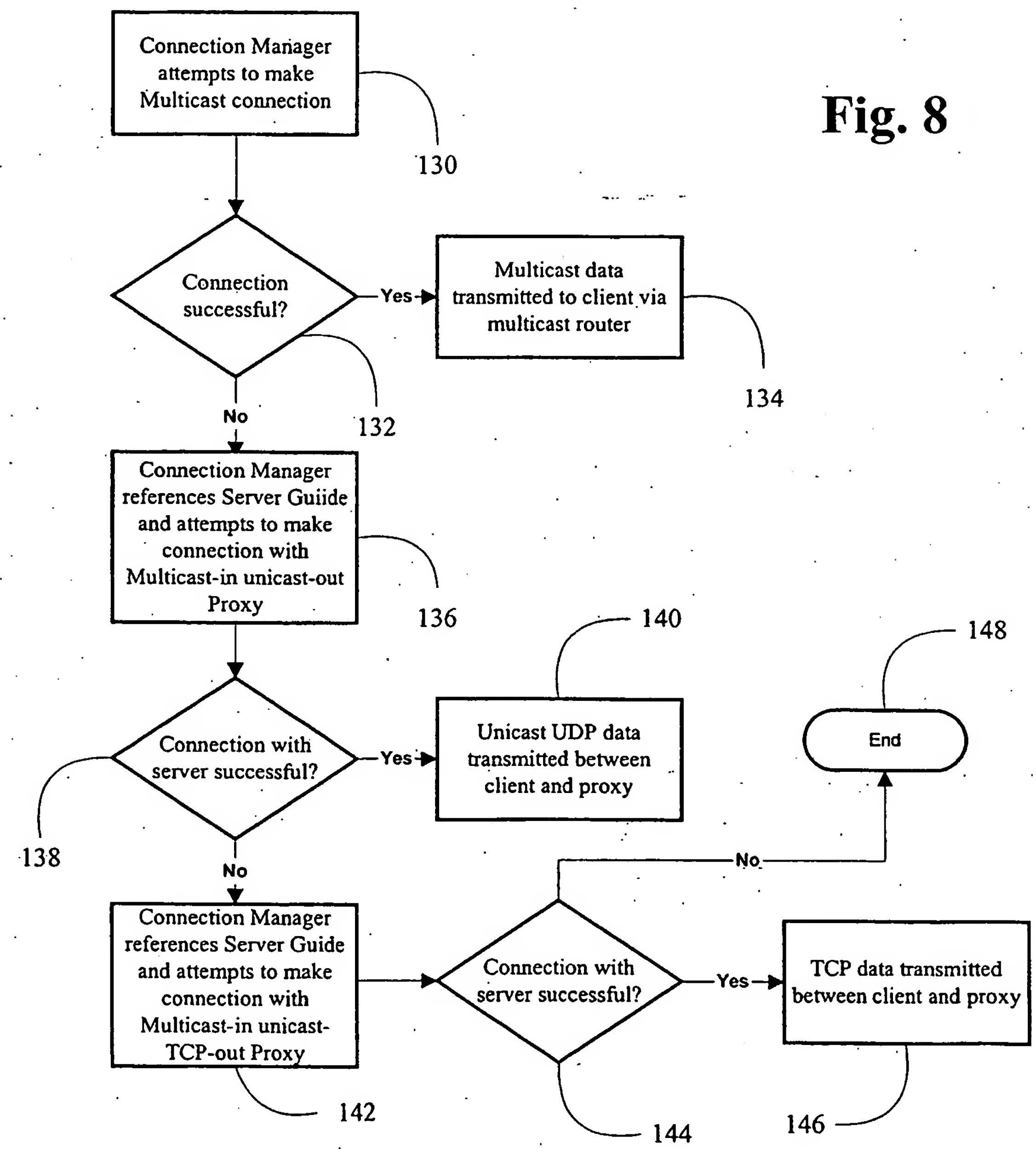

Connection Manager attempts to make Multicast connection
130
Connection successful?
Yes
Multicast data transmitted to client via multicast router
134
No
132
Connection Manager references Server Guiide and attempts to make connection with Multicast-in unicast-out Proxy
136
140
148
Connection with server successful?
Yes
Unicast UDP data transmitted between client and proxy
End
138
No
No
Connection Manager references Server Guide and attempts to make connection with Multicast-in unicast-TCP-out Proxy
Connection with server successful?
Yes
TCP data transmitted between client and proxy
142
144
146

Fig. 8

Fig. 9

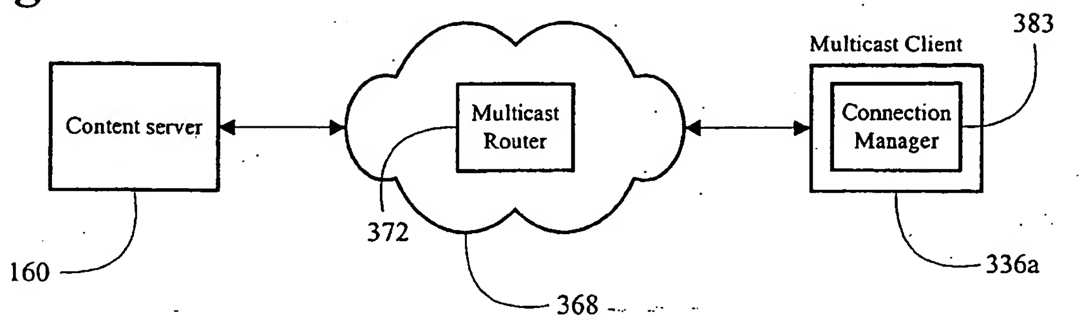


Fig. 10

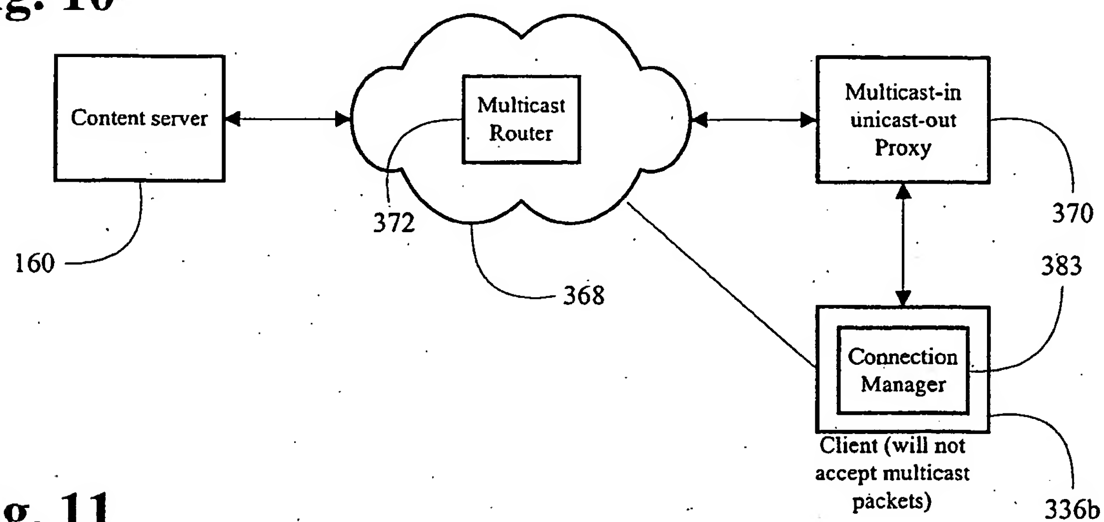
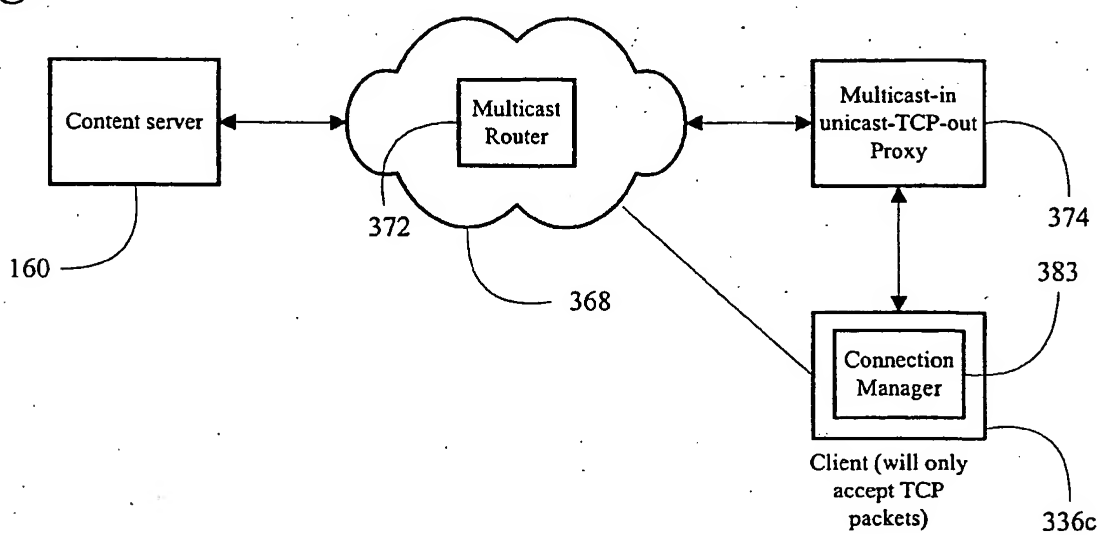


Fig. 11



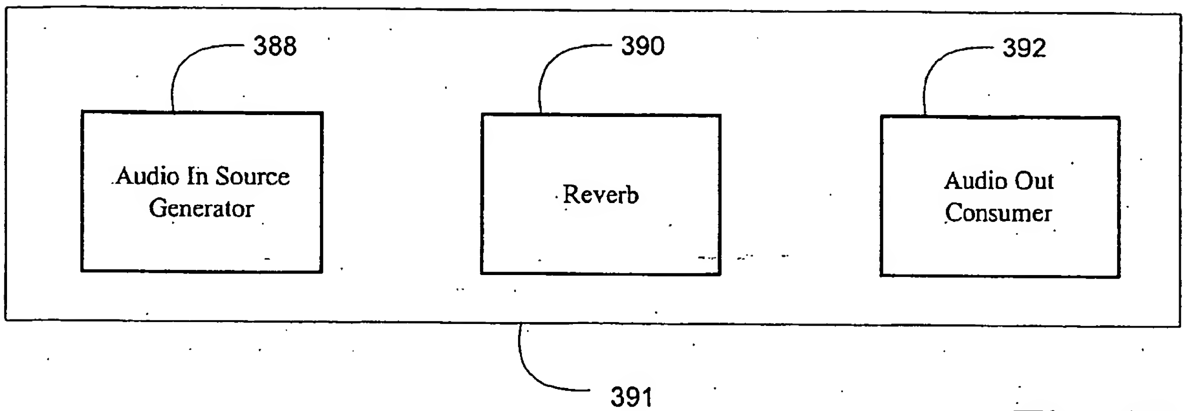

388
390
392
Audio In Source Generator
Reverb
Audio Out Consumer
391

Fig. 12

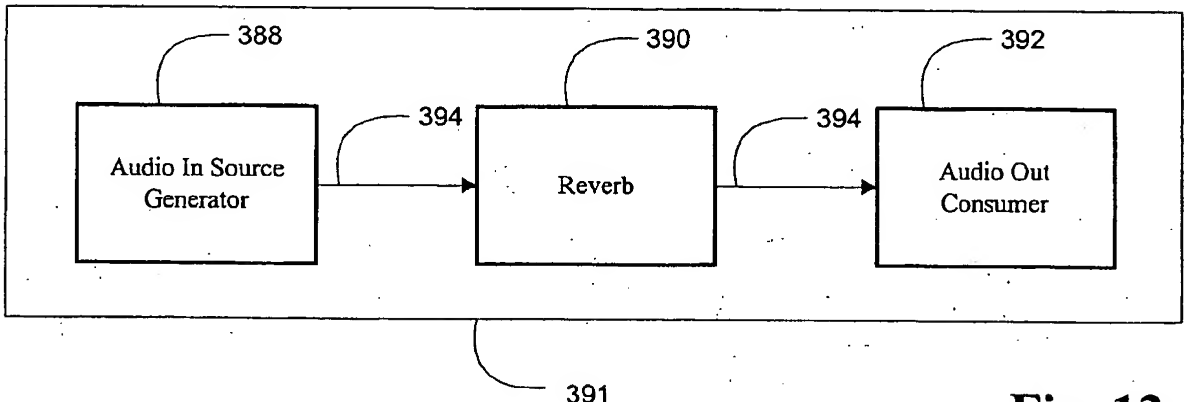

388
390
392
394
394
Audio In Source Generator
Reverb
Audio Out Consumer
391

Fig. 13

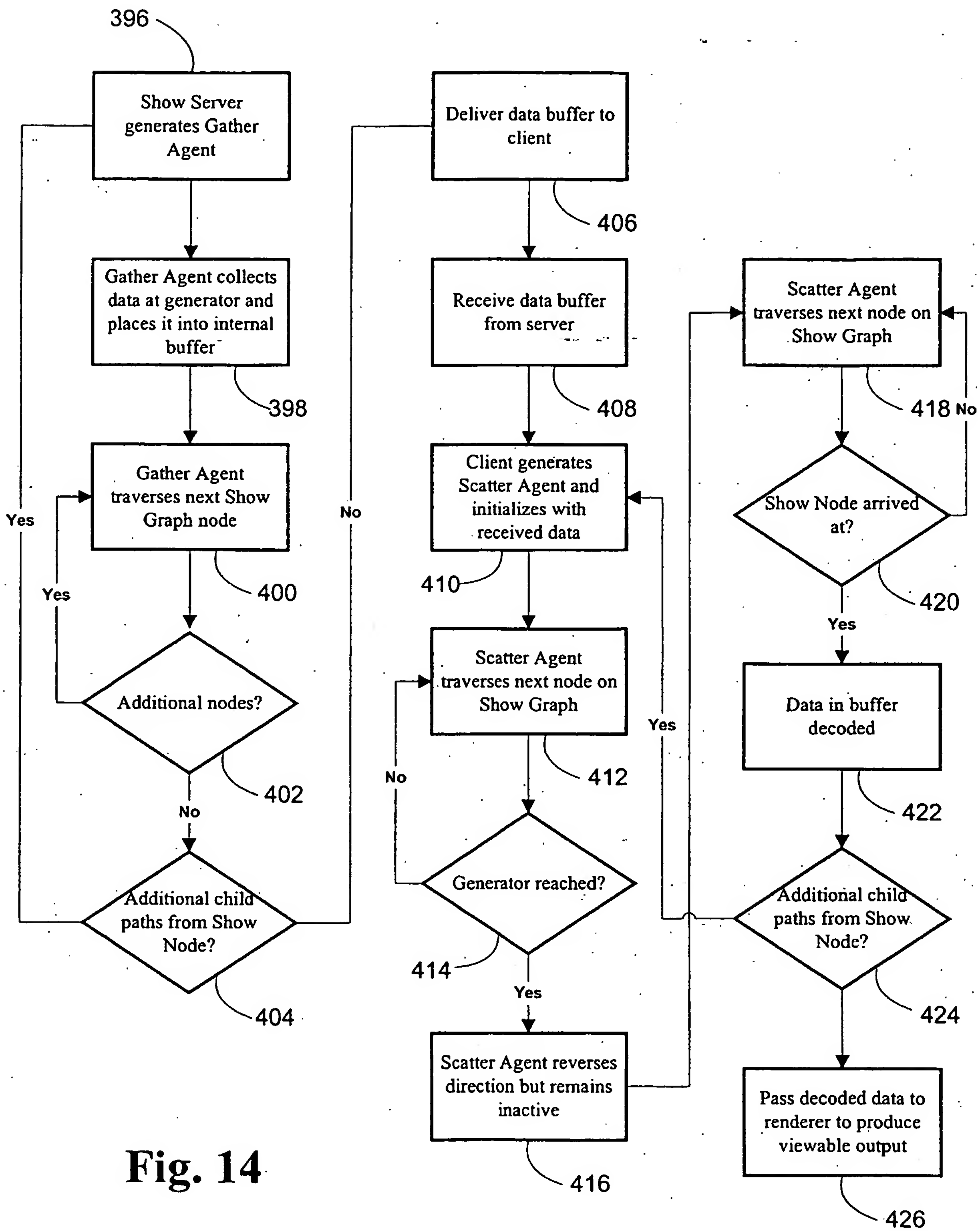

396
Show Server generates Gather Agent
Gather Agent collects data at generator and places it into internal buffer
398
Gather Agent traverses next Show Graph node
400
Yes
Additional nodes?
402
No
Additional child paths from Show Node?
404
Yes
No
Deliver data buffer to client
406
Receive data buffer from server
408
Client generates Scatter Agent and initializes with received data
410
Scatter Agent traverses next node on Show Graph
412
No
Generator reached?
414
Yes
Scatter Agent reverses direction but remains inactive
416
Scatter Agent traverses next node on Show Graph
418
No
Show Node arrived at?
420
Yes
Data in buffer decoded
422
Additional child paths from Show Node?
424
Yes
Pass decoded data to renderer to produce viewable output
426

Fig. 14

Fig. 15

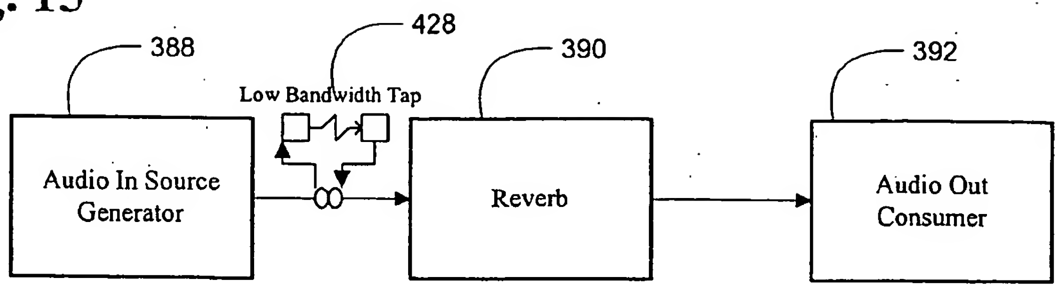


Fig. 16

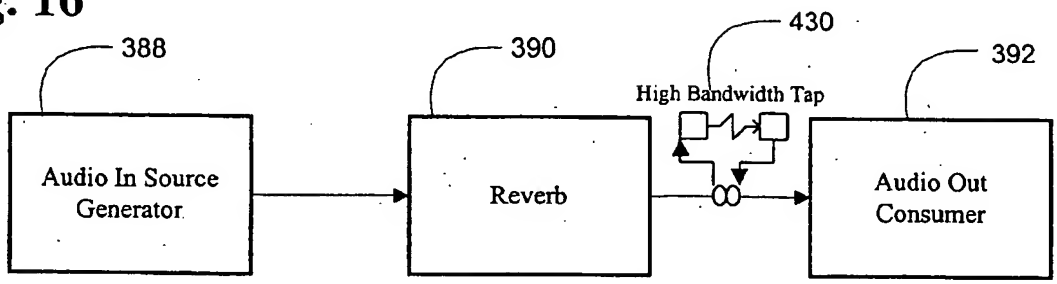
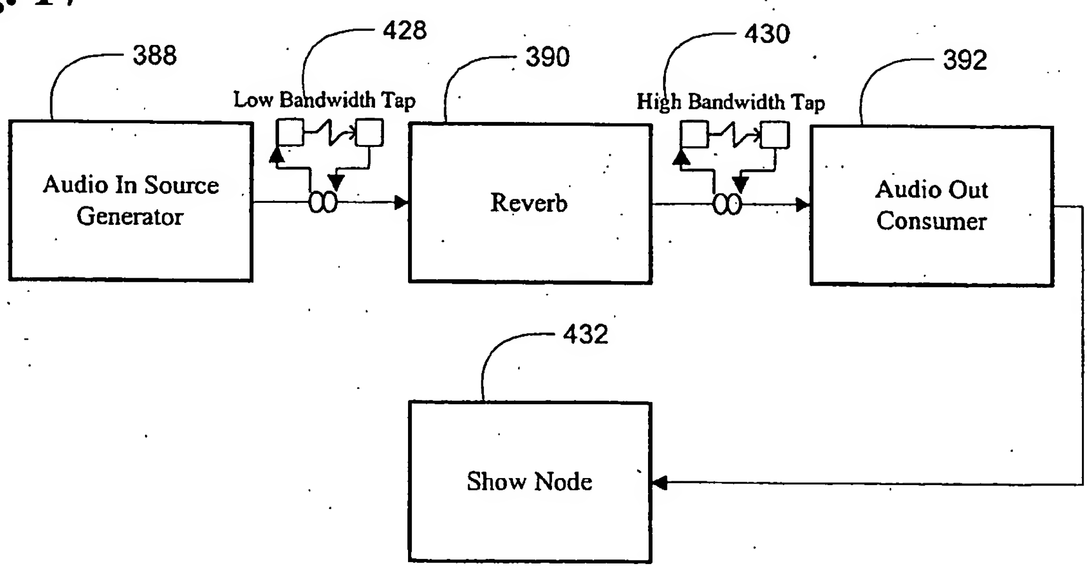


Fig. 17



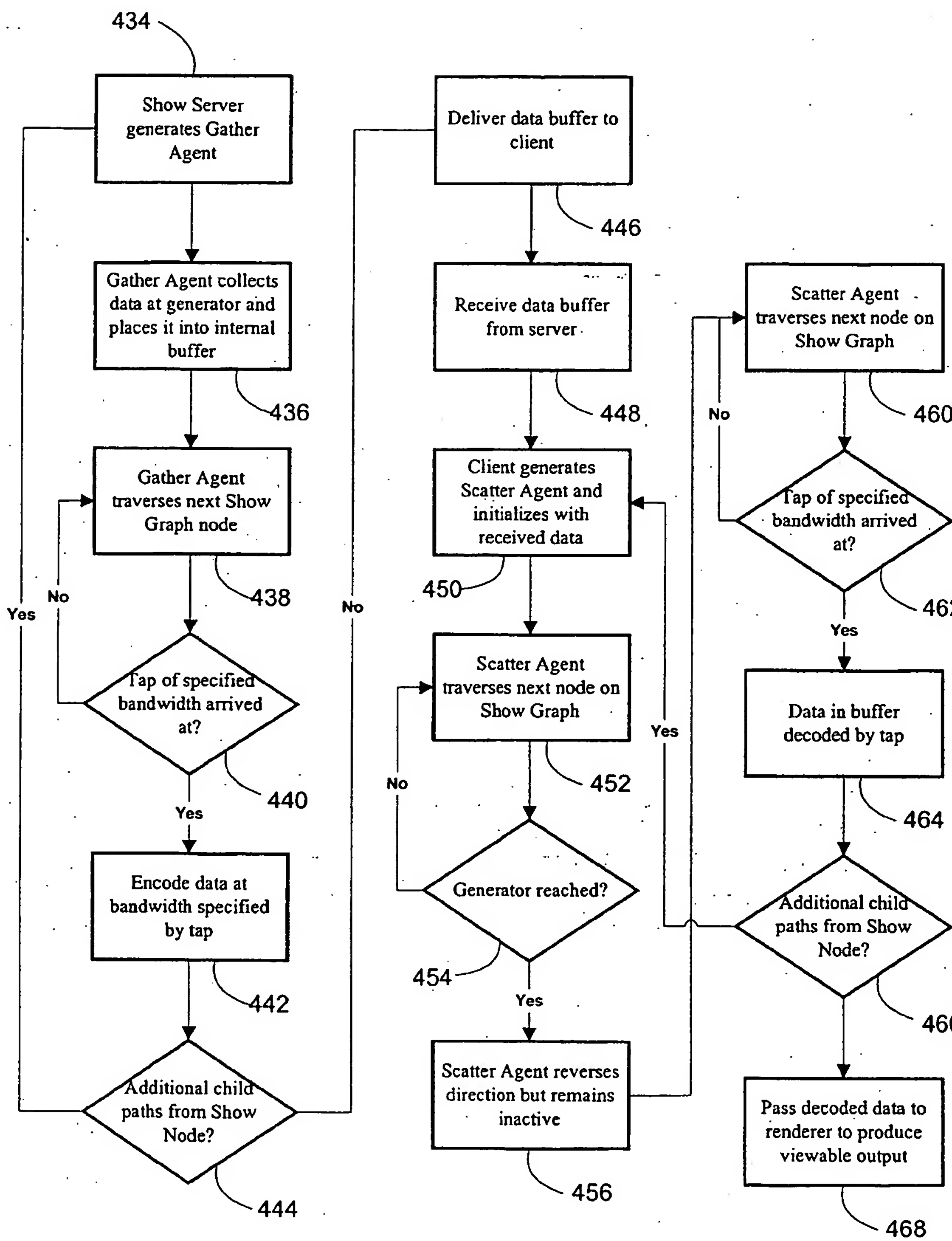

434
Show Server generates Gather Agent
Gather Agent collects data at generator and places it into internal buffer
436
Gather Agent traverses next Show Graph node
438
Tap of specified bandwidth arrived at?
440
No
Yes
Encode data at bandwidth specified by tap
442
Additional child paths from Show Node?
444
Yes
No
Deliver data buffer to client
446
Receive data buffer from server
448
Client generates Scatter Agent and initializes with received data
450
Scatter Agent traverses next node on Show Graph
452
Generator reached?
454
No
Yes
Scatter Agent reverses direction but remains inactive
456
Scatter Agent traverses next node on Show Graph
460
Tap of specified bandwidth arrived at?
462
No
Yes
Data in buffer decoded by tap
464
Additional child paths from Show Node?
466
Yes
Pass decoded data to renderer to produce viewable output
468

Fig. 18

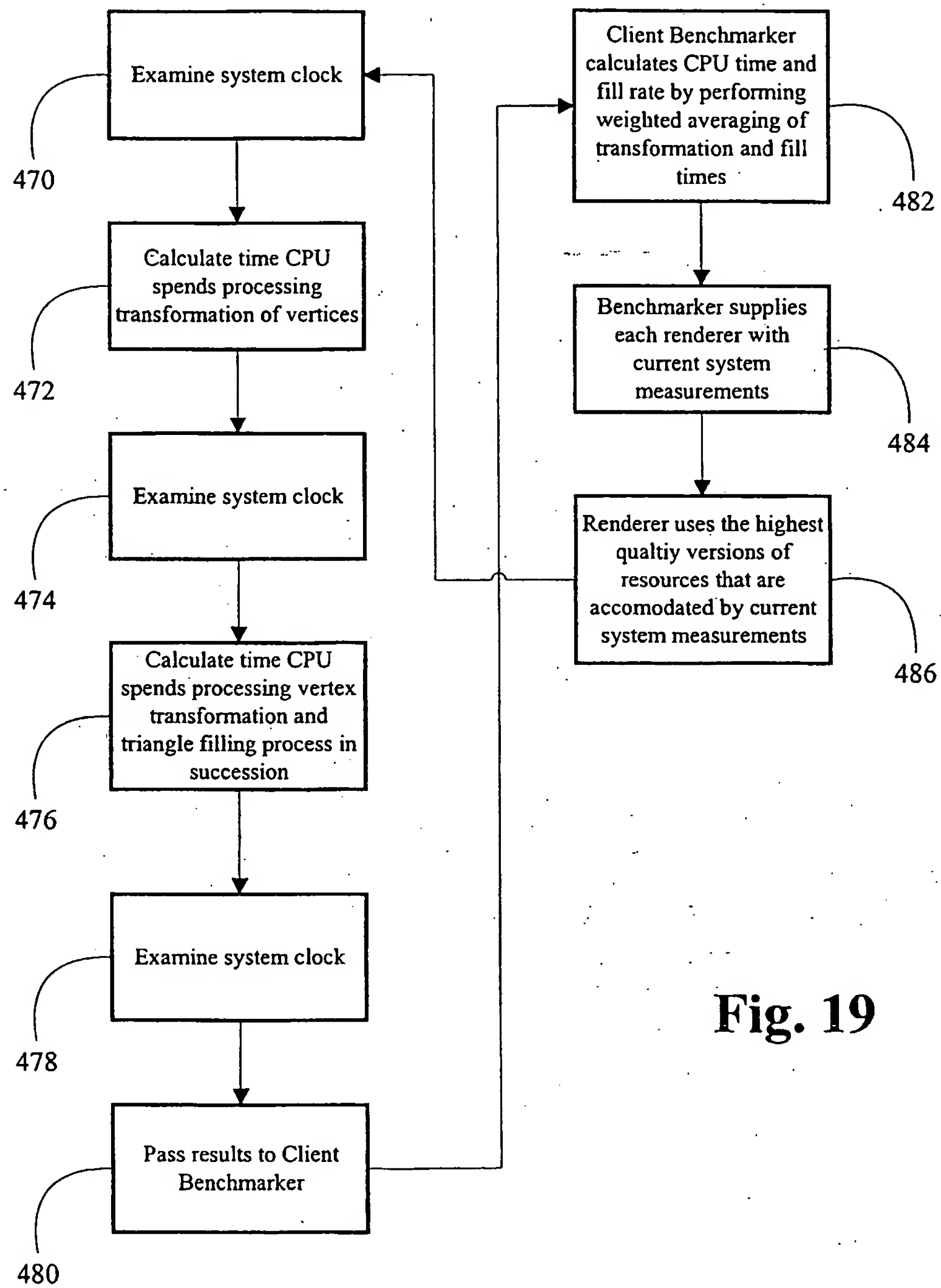

Examine system clock
470
Calculate time CPU spends processing transformation of vertices
472
Examine system clock
474
Calculate time CPU spends processing vertex transformation and triangle filling process in succession
476
Examine system clock
478
Pass results to Client Benchmarker
480
Client Benchmarker calculates CPU time and fill rate by performing weighted averaging of transformation and fill times
482
Benchmarker supplies each renderer with current system measurements
484
Renderer uses the highest qualtiy versions of resources that are accomodated by current system measurements
486

Fig. 19

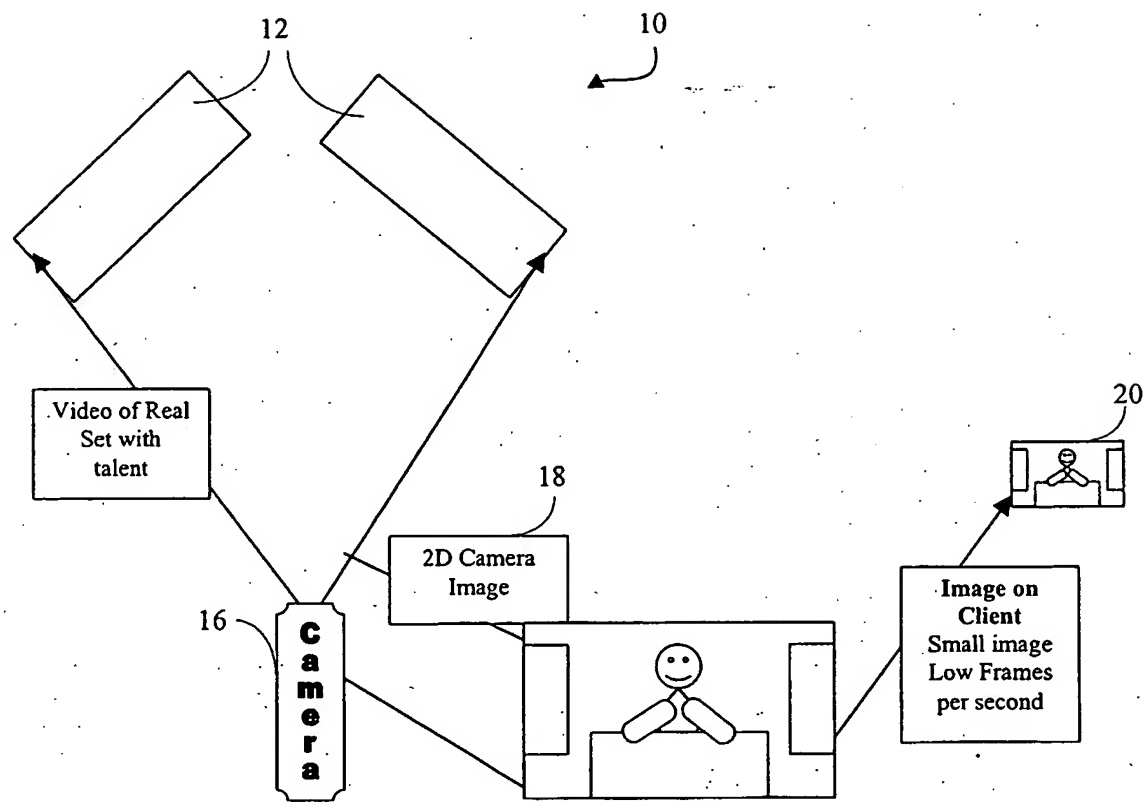

12
10
Video of Real Set with talent
20
18
2D Camera Image
16
Camera
Image on Client Small image Low Frames per second

FIG. 20

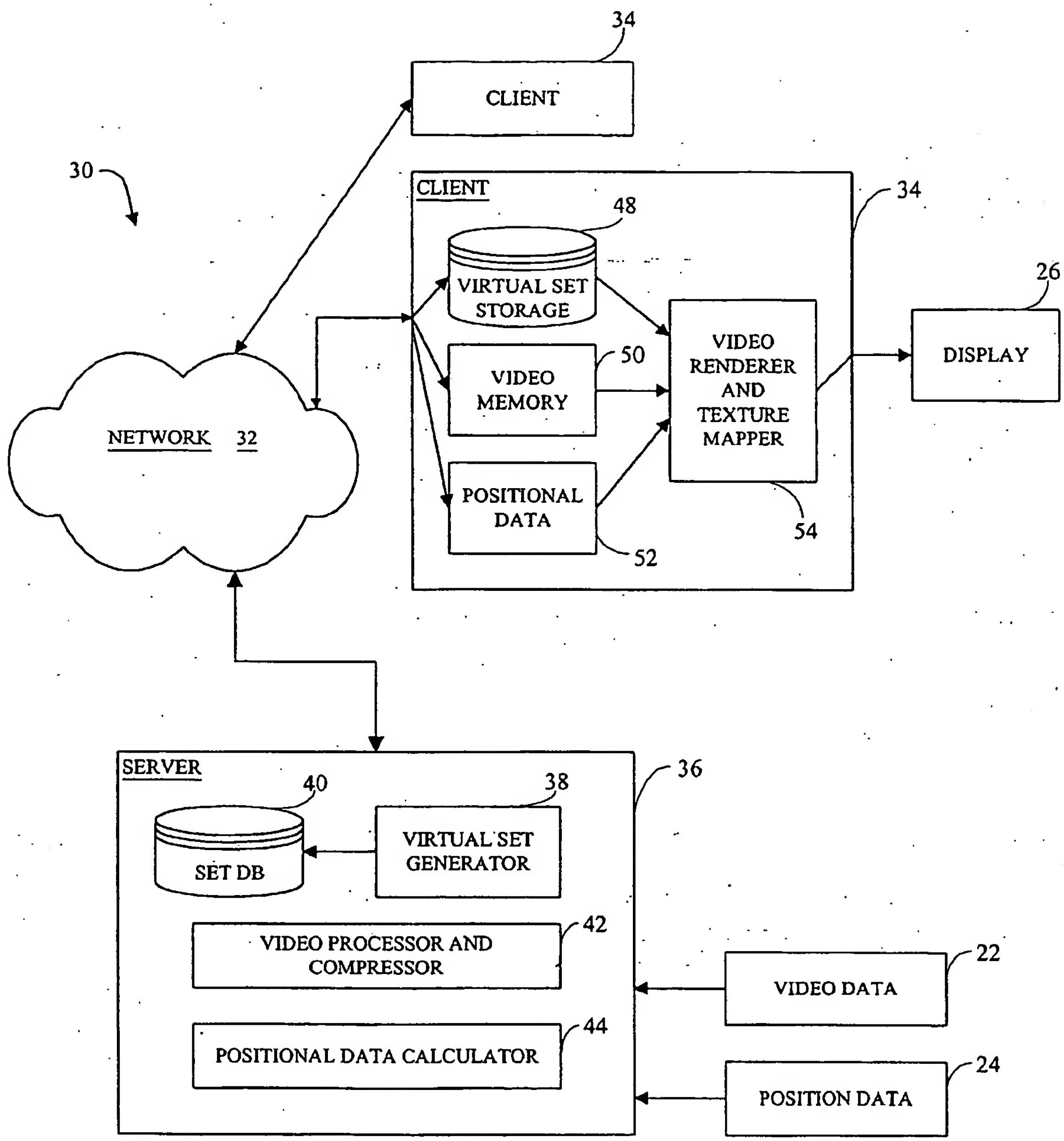
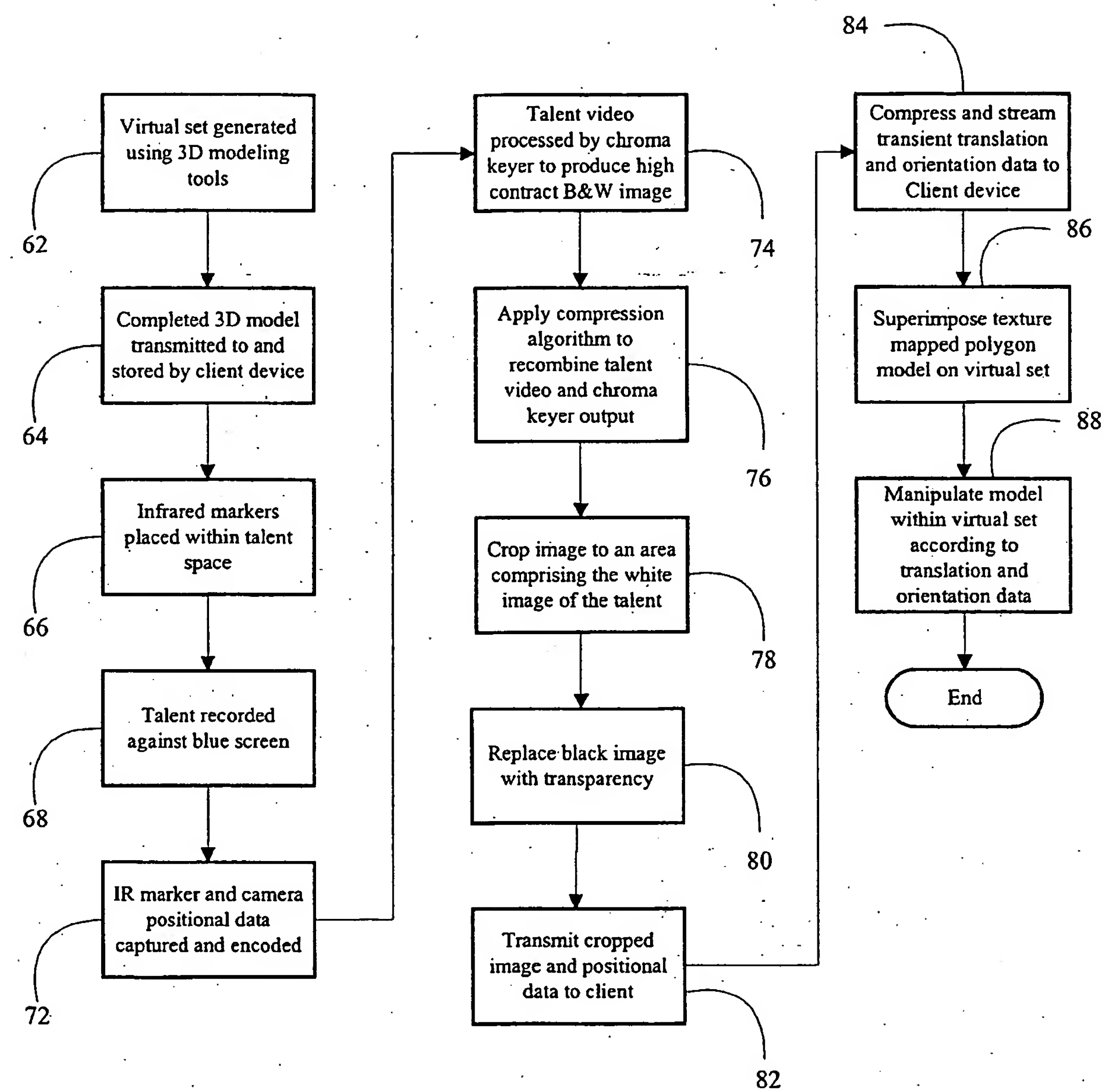

34
CLIENT
30
CLIENT
34
48
VIRTUAL SET STORAGE
26
DISPLAY
VIDEO MEMORY
50
VIDEO RENDERER AND TEXTURE MAPPER
NETWORK 32
POSITIONAL DATA
52
54
SERVER
36
40
38
SET DB
VIRTUAL SET GENERATOR
VIDEO PROCESSOR AND COMPRESSOR
42
22
VIDEO DATA
POSITIONAL DATA CALCULATOR
44
24
POSITION DATA

FIG. 21



Virtual set generated using 3D modeling tools
62
Completed 3D model transmitted to and stored by client device
64
Infrared markers placed within talent space
66
Talent recorded against blue screen
68
IR marker and camera positional data captured and encoded
72
Talent video processed by chroma keyer to produce high contract B&W image
74
Apply compression algorithm to recombine talent video and chroma keyer output
76
Crop image to an area comprising the white image of the talent
78
Replace black image with transparency
80
Transmit cropped image and positional data to client
82
84
Compress and stream transient translation and orientation data to Client device
86
Superimpose texture mapped polygon model on virtual set
88
Manipulate model within virtual set according to translation and orientation data
End

FIG. 22

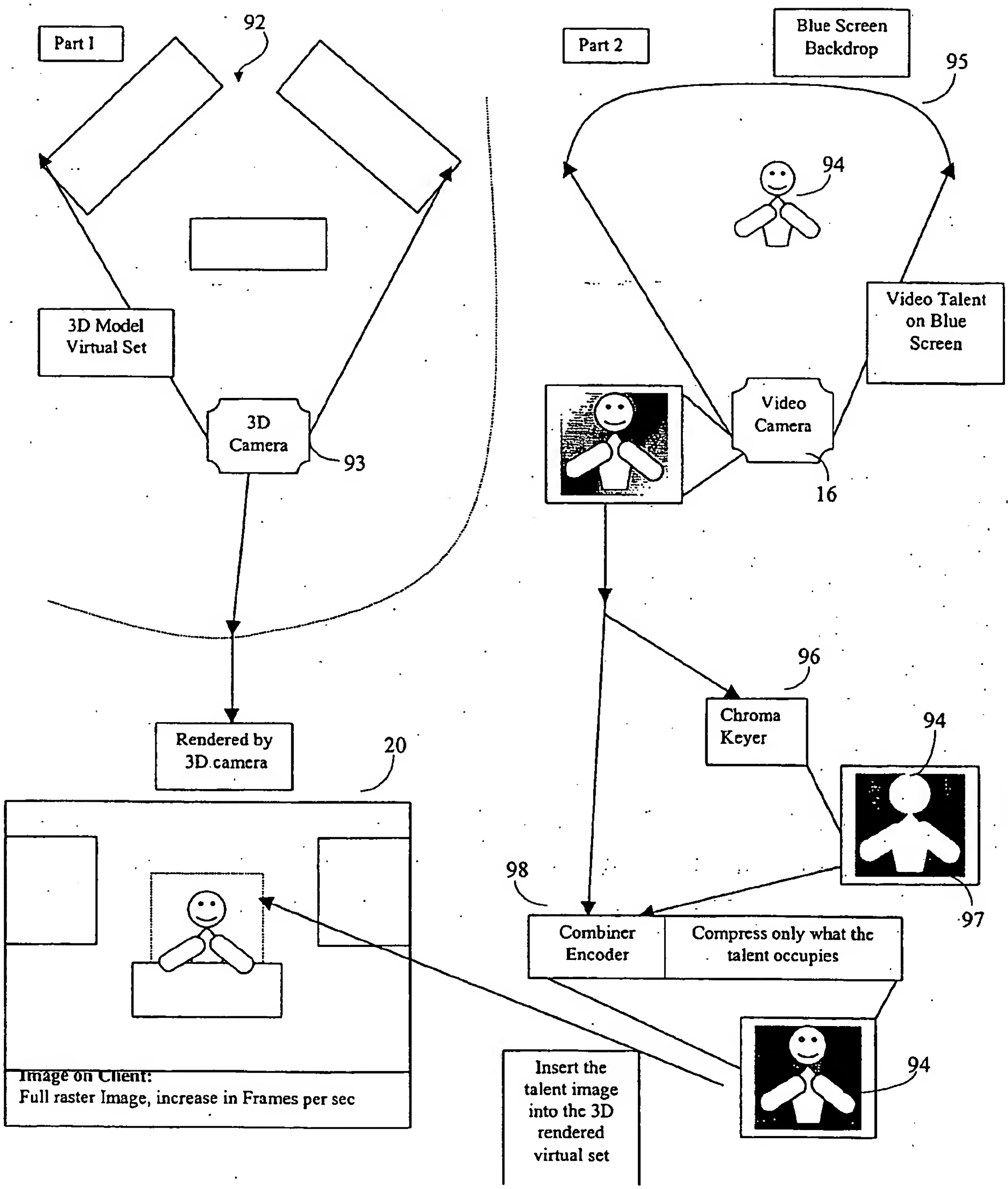

Part I
92
3D Model
Virtual Set
3D
Camera
93
Rendered by
3D camera
20
Image on Client:
Full raster Image, increase in Frames per sec
Part 2
Blue Screen
Backdrop
95
94
Video Talent
on Blue
Screen
Video
Camera
16
96
Chroma
Keyer
94
97
98
Combiner
Encoder
Compress only what the
talent occupies
Insert the
talent image
into the 3D
rendered
virtual set
94

FIG. 23

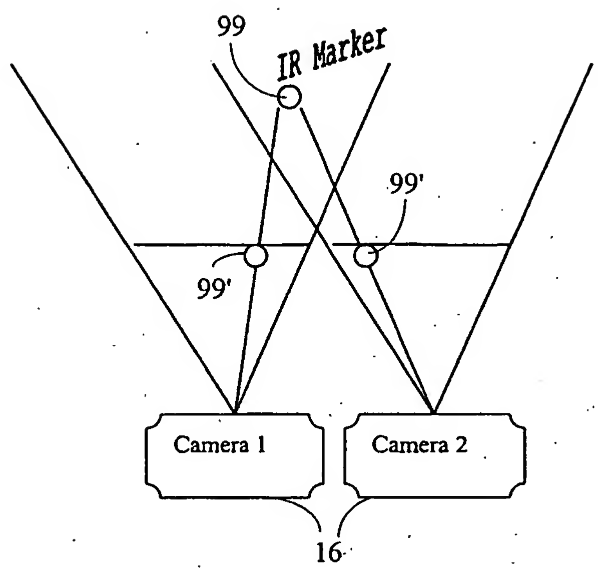


FIG. 24

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/02224

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 15/16
US CL : 709/201,, 218, 226, 231
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 709/201,, 218, 226, 231

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

STN: bandwidth video, camera, crop, mat, angle view, table, packet, missing

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,951,694 A (CHOQUIER et al.) 14 September 1999, abstract	1-28
Y	US 5,958,012 A (BATTAT et al.) 28 September 1999, abstract	1-28
A	US 5,719,854 A (CHOUDHURY et al.) 17 February 1998, abstract	1-28
A	US 5,905,877 A (GUTHRIE et al.) 18 May 1999, abstract	1-28
Y	US 5,850,352 A (MOEZZI et al.) 15 December 1998, abstract	29-44
Y	US 5,956,039 A (WOODS et al.) 21 September 1999, abstract	29-44

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
23 APRIL 2001	31 MAY 2001

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	ROBERT HARRELL James R. Matthews
Facsimile No. (703) 305-3230	Telephone No. (703) 305-9692

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/02224

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y, P	US 6,084,979 A (KANADE et al.) 04 July 2000, abstract	29-44
Y, P	US 6,029,175 A (CHOW et al.) 22 February 2000, abstract	1-28, 45-59
Y, P	US 6,047,323 A (KRAUSE) 04 April 2000, abstract	1-28, 45-59
A	US 5,805,804 A (LAURSEN et al.) 08 September 1998, abstract	45-59
Y	US 5,949,772 A (SUGIKAWA et al.) 07 September 1999, abstract	45-69
Y	US 5,999,940 A (RANGER) 07 December 1999, abstract	45-69
Y, P	US 6,073,250 A (LUBY et al.) 06 June 2000, abstract	60-81
Y, P	US 6,081,909 a (LUBY et al.) 27 June 2000, abstract	60-81
Y, P	US 6,081,918 A (SPIELMAM) 27 June 2000, abstract	60-81

INTERNATIONAL SEARCH REPORT	International application No. PCT/US01/02224

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

Please See Extra Sheet.

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☒ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/02224

BOX II. OBSERVATIONS WHERE UNITY OF INVENTION WAS LACKING
This ISA found multiple inventions as follows:

This application contains the following inventions or groups of inventions which are not so linked as to form a single inventive concept under PCT Rule 13.1. In order for all inventions to be searched, the appropriate additional search fees must be paid.

Group I, claim(s) 1-29, are drawn to production process that depdends on bandwidth and a visual of the network.
Group II, claim(s) 29-44, are drawn to camera position data in a video stream.
Group III, claim(s) 45-59, are drawn to determining which server to use with a table of servers.
Group IV, claim(s) 60-81, are drawn to resending a missing packet from a sequence based on time.
The inventions listed as Groups I-IV do not relate to a single inventive concept under PCT Rule 13.1 because, under PCT Rule 13.2, they lack the same or corresponding special technical features for the following reasons: Group I involves presentation selection based on bandwidth while Group II invovles camera location in a video stream while Group III involves selection of a server, and lastly, Group IV invovles a determination of how to resend a missing packet in a sequence.